AD-A111 144   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH   SCHOO--ETC   F/G 17/5
ALTERNATIVES TO AN EXTENDED KALMAN FILTER FOR TARGET IMAGE TRAC--ETC(U)
DEC 81   P R LEUTHAUSER
UNCLASSIFIED   AFIT/GA/AA/81D-8   NL

1 OF 3
AD
A111144

LEVEL II

AD A111144

S FEB 19 1982 D

E

DTIC FILE COPY

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

This document has been approved for public release and sale; its distribution is unlimited.

82 02 18 135

LEVEL II ①

AFIT/GA/AA/81D-8

ALTERNATIVES TO AN EXTENDED KALMAN FILTER FOR TARGET IMAGE TRACKING

THESIS

AFIT/GA/AA/81D-8

Paul R. Leuthauser
Capt USAF

Approved for public release; distribution unlimited

AFIT/GA/AA/81D-8

ALTERNATIVES TO AN EXTENDED KALMAN
FILTER FOR TARGET IMAGE TRACKING

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Accession For
NTIS GRA&I ☒
DTIC ☐
Unann ☐
Just
By
D
Dis
A

by
Paul R. Leuthauser
Capt USAF
Graduate Astronautical Engineering
December 1981

Approved for public release; distribution unlimited

Preface

This study was part of a continuing effort to design a missile tracker for one of the Air Force Weapons Laboratory's laser weapons using modern estimation techniques. Earlier studies demonstrated the advantages of using an extended Kalman filter which uses direct infrared sensor data. My efforts were intended to extend this design effort by synthesizing and testing four alternative filters to the extended Kalman filter.

I would like to express my thanks to my advisor Dr. Peter S. Maybeck for his motivation, advice, and time, and to an understanding typist Julie Meyer. Finally, a special thanks to my wife and daughter, Janine and Tyree, for their good humor.

Paul R. Leuthauser

Table of Contents

Page

Preface ii

List of Figures v

List of Tables. vii

List of Symbols viii

Abstract. x

I. Introduction 1

  Background. 1

  Problem 2

  Scope 2

  Assumptions 3

    FLIR. 3

    Closed-Loop 3

    Basic Model 3

  Best Performance. 4

  Approach. 4

II. Previous Investigations. 6

  Benign Tracking Task. 6

  Air-to-air Missile Tracking Task. 12

    Improved Target Intensity Model 12

    Improved Dynamics Model 14

  Baseline Extended Kalman Filter 15

III. Alternative Filters. 17

  Extended Kalman Filter with Bias Correction Term. . . . 18

  Constant Gain Filters 21

  Statistically Linearized Filter 22

    Evaluating $\widehat{\underline{h}[\underline{x}(t_i),t_i]}$. 25

    Evaluating $\widehat{\underline{h}[\underline{x}(t_i),t_i]\underline{x}^T(t_i)}$. 30

    Algorithm Summary 34

IV. Analysis Methods 36

  Monte Carlo Study 36

  Computational Burden Analysis 43

V. Results. 44

  Best Performance. 45

Page
Imperfect Initial Conditions. 49
Low Signal-to-noise Ratio Performance 54
Incorrect Filter Modelling Performance. 54
Statistically Linearized Filter 56
Computational Burden. 61
VI. Conclusions & Recommendations 65
Conclusions 65
Extended Kalman Filter with Bias
Correction Term. 65
Constant Gain Extended Kalman Filter 65
Statistically Linearized Filter. 65
Recommendations 66

Bibliography 68

Appendix A 70

Appendix B 82

Appendix C 85

Appendix D 89

Appendix E 127

Appendix F 191

Appendix G 249

List of Figures

Figure Page

1 Circular Equal-intensity Contours. 7

2 Elliptical Equal-intensity Contours. 13

3 Region for Evaluating P_ℓ 29

4 Realizations of $[x_{peak}(t_i), y_{peak}(t_i)]_\ell$ . . 31

5 Evaluating P_ℓ. 32

6 Access of Values for $\underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_\ell$ 33

7 Monte Carlo Study. 38

8 Variance Convergence 40

9 Target Intensity Mismatch Performance-Extended Kalman Filter 57

10 Target Intensity Mismatch Performance-Constant Gain EKF. 58

A-1 Truth Model Reference Frame. 74

A-2 Instensity Image Orientation 77

E-1 Performance Plot - Case E-1. 131

E-2 Performance Plot - Case E-2. 135

E-3 Performance Plot - Case E-3. 143

E-4 Performance Plot - Case E-4. 151

E-5 Performance Plot - Case E-5. 155

E-6 Performance Plot - Case E-6. 159

E-7 Performance Plot - Case E-7. 167

Figure Page

E-9 Performance Plot - Case E-9. 171
E-10 Performance Plot - Case E-10 175
E-11 Performance Plot - Case E-11 179
E-12 Performance Plot - Case E-12 183
E-13 Performance Plot - Case E-13 187
F-1 Performance Plot - Case F-1. 195
F-2 Performance Plot - Case F-2. 199
F-3 Performance Plot - Case F-3. 207
F-4 Performance Plot - Case F-4. 215
F-5 Performance Plot - Case F-5. 219
F-6 Performance Plot - Case F-6. 223
F-9 Performance Plot - Case F-9. 229
F-10 Performance Plot - Case F-10 233
F-11 Performance Plot - Case F-11 237
F-12 Performance Plot - Case F-12 241
F-13 Performance Plot - Case F-13 245
G-1 Performance Plot - Case G-1. 252
G-2 Performance Plot - Case G-2. 254
G-3 Performance Plot - Case G-3. 256
G-4 Performance Plot - Case G-4. 258
G-6 Performance Plot - Case G-6. 260
G-7 Performance Plot - Case G-7. 262

List of Tables

Table — Page

I Truth Model Trajectories. 42

II Performance Comparison, Ideal Conditions. 46

III Performance Comparison, Ideal Conditions. 47

IV Bias Correction Term and Filter Measurement Comparison. 48

V Initial Condition Mismatch Recovery Comparison. . . . 51

VI Filter Acquisition with Bias Correction Term. 53

VII Performance Comparison, Low S/N Ratio 55

VIII Statistically Linearized Performance with Varying $f_{\underset{\sim}{x}|\underset{\sim}{z}}$ 60

IX Computational Burden. 62

C-I Turn Trajectory $\underline{K}^T_{ss}$. 87

C-II Constant Velocity Trajectory $\underline{K}^T_{ss}$ 88

List of Symbols

t, τ	time
$\underline{x}$	State vector
$\hat{\underline{x}}$	Filter state estimate vector
$\underline{P}$	Covariance Matrix
$\underline{Q}_d$	Discrete process noise strength matrix
I	Maximum intensity
$\underline{R}$	Covariance of measurement noise matrix
$\underline{z}$	General measurement vector
$\underline{Z}$	Time history of measurements vector
$\underline{\Phi}$	State transition matrix
S/N	Signal to noise ratio
ξ_x	Dummy variable
ξ_y	Dummy variable
ξ	Dummy variable
ρ	Dummy variable
σ	RMS, variance, or standard deviation
$E\{\cdot\}$	Expected value operator
$E\{\cdot\|\cdot\}$	Conditional expected value operator
$\underset{\sim}{\underline{z}}(t_i,\omega_i)$	Random variable defined for a fixed time, t_i, and stochastic process sample, ω_i
$\delta(t)$, $\delta(\tau)$	Delta function
$[\cdot]^T$	Matrix transpose
$[\cdot]^{-1}$	Matrix inverse

Superscripts

$\dot{x}$	Time derivative
$\hat{x}$	Estimate
t_i^+	After update
t_i^-	Before update

Subscripts

I_F	Filter
I_T	Truth value
t_i	Current sample time
t_{i-1}	Previous sample time
x_0	Initial value
x_e	Inertial reference frame

Underline

$\underline{x}$	Vector or matrix
$\underset{\sim}{x}$	Stochastic process

Abstract

Four alternative filters are compared to an extended Kalman filter (EKF) algorithm for tracking a distributed (elliptical) source target in a closed loop tracking problem, using outputs from a forward looking (FLIR) sensor as measurements. These were 1) an EKF with (second order) bias correction term, 2) a constant gain EKF, 3) a constant gain EKF with bias correction term, and 4) a statistically linearized filter. Estimates are made of both actual target motion and of apparent motion due to atmospheric jitter. These alternative designs are considered specifically to address some of the significant biases exhibited by an EKF due to initial acquisition difficulties, unmodelled maneuvering by the target, low signal-to-noise ratio, and real world conditions varying significantly from those assumed in the filter design (robustness). Filter performance was determined with a Monte Carlo study under both ideal and non ideal conditions for tracking targets on a constant velocity cross range path, and during constant acceleration turns of 5G, 10G, and 20G. The EKF with bias correction term demonstrated nearly identical performance to the EKF, with a cost of four times the computational burden. The constant gain EKF ideal performance was not as good, but it was more robust and required only one-tenth the computational burden. The statistically linearized filter was unsuccessful due to the particular discretization approximation used in evaluating conditional expectations inherent in its structure.

I. Introduction

Background

Recent years have seen the advent of high energy laser weapons. They offer the distinct advantage of being able to deliver lethal amounts of energy to a target essentially instantaneously. An efficient implementation of this weapon system maintains the laser beam on only a specific part of the vehicle until it is destroyed or incapacitated, rather than using enormous amounts of energy to "paint" the entire target vehicle with laser energy. Such an implementation requires both precise pointing of the laser and accurate tracking of the target. It is the tracking portion of this system that is the subject of this study.

Previous studies (Refs 13; 6) have shown the advantage of using an extended Kalman filter as a tracker because of its ability to extract more information from sensor measurements than does a currently used correlation algorithm (Refs 15; 16; 17; 18). First, the statistical characteristics of the atmospheric effect on the infrared wavefront into an infrared sensor can be modelled. This allows the filter to separate true target motion from atmospheric jitter. Second, the dynamics model imbedded in the extended Kalman filter gives it the ability to predict future target position. Thus, for a sample data pointing and tracking system, the tracker is able to command the pointer to future target positions.

Problem

The Air Force Weapons Laboratory is currently using a Forward Looking Infrared (FLIR) sensor in conjunction with a tracking system to provide accurate target position estimates. Specifically, the ability is needed to track air-to-air missiles at close range to accuracies considerably better than one milliradian standard deviation. In this dynamic scenario, the validity of a first order extended Kalman filter using FLIR outputs is questionable because of high maneuverability and the low signal-to-noise ratio environment. As a result, alternative filters are developed in this study which also exploit the same information as the extended Kalman filter.

Scope

In an effort to provide precise target state estimates, several filters are developed, evaluated, and compared as alternatives to an extended Kalman filter. These filters include:

1) Extended Kalman Filter with Bias Correction Term
2) Constant Gain Extended Kalman Filter
3) Constant Gain Extended Kalman Filter with Bias Correction Term
4) Statistically Linearized Filter

Filter performance is evaluated by tracking simulated trajectories which include both typical and worst case missile dynamics. Such a worst case situation is the one alluded to previously, that of tracking an accelerating missile in a cluttered background. In addition, filter robustness to imperfect initial conditions and modelling errors is addressed. Finally, filter performance is weighed against computational burden.

Assumptions

FLIR. The FLIR, as described in (Ref 6:p4), provides a frame of data every thirtieth of a second for use by the filter, thus defining a 30 Hz data sample rate. For this study, an 8-by-8 array of pixels extracted from a much larger FLIR frame of pixels constitutes a single measurement array. Computational and storage limitations are responsible for reducing the measurement array to this smaller "tracking window". Finally, biases in alternating rows of FLIR data (Ref 6:p5) were assumed to be compensated prior to the data reaching the filter.

Closed Loop. The tracker is assumed to be operating within a closed loop pointing and tracking system, where the laser pointing system works perfectly. Thus, the system can point exactly where the tracker commands it within each sample period, which implies that the pointing system's settling time is less than one data sample period (1/30 sec). This dictates the requirement for a good dynamics propagation model within the filter.

Basic Model. Throughout this study, the filter algorithms are based on a linear dynamics model and a nonlinear measurement model. The dynamics model is detailed in Section II, and it is implemented the same in all the algorithms. It is the method of dealing with the nonlinear measurement that distinguishes each of the filters.

Best Performance

Simulations to determine each filter's best performance were conducted under rather ideal conditions. First, the signal-to-noise ratio, a ratio of target intensity to rms background noise intensity on the FLIR output, was held high, but at a physically realistic value. Second, artificial a prior knowledge of when maneuvers began was used to tune the filter. Adaptive estimation techniques have been developed to provide filter self-tuning (Ref 6:p122;12), but this added complexity would only cloud the basic filter structures being evaluated. Third, perfect knowledge of initial conditions was assumed in the early analyses so that filter performance did not suffer from transients which can arise with imperfect initial conditions. This study also addresses the performance degradation due to imperfect filter modelling.

Approach

To establish a foundation for this study, Section II relates a history of previous investigations. An extended Kalman filter is derived which serves as the baseline tracker for this study. The extended Kalman filter provides levels of performance and computational burden to which alternative filter forms may be compared.

Next, in Section III, the alternative filters are developed. The addition of bias correction terms provides second order information previously ignored. Often the extended Kalman filter produces biased estimates from harsh nonlinearities because they are based on only a first order Taylor series representation. Constant gain filters promise not only lower computational burden, but they may also demonstrate better robustness characteristics. (Ref 19:p721) Finally, the statistically linearized filter does not depend upon a truncated Taylor series representation, but rather relies upon a statistical approach to minimizing the error in approximating the nonlinearity. Such an approach works well when the filter's confidence in its estimate is low enough so that the true state value may lie outside the linearized region.

To complete the study, Section IV describes the analysis methods used, Section V presents the results of this research effort, and Section VI presents conclusions and recommendations.

II. Previous Investigations

Laser pointing and tracking is a topic of continuing research at the AF Institute of Technology. This section contains a summary of the prior work which forms the basis for this study.

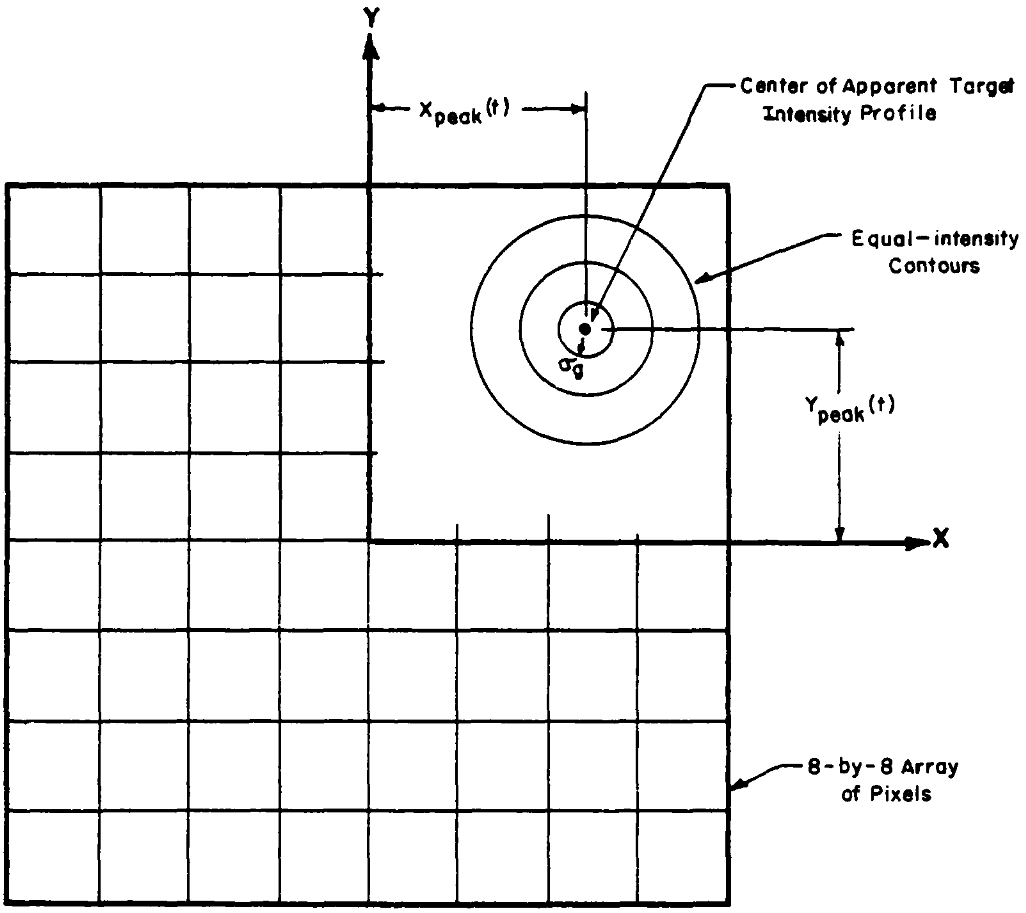
Benign Tracking Task

As originally pursued by Mercier (Ref 13), the accurate tracking task was directed at long range targets, where even large targets appear as point sources of infrared radiation. This simplified modelling the target's intensity pattern on the FLIR image plane. Also, such targets appear to the tracker to exhibit benign dynamics, which kept the dynamics model simple.

By considering the physics of wave propagation and optics (Ref 14:p15), Mercier modelled the target's intensity pattern on the FLIR image plane as a bivariate Gaussian function with circular equal intensity contours as shown in Fig 1. The model was

$$I_{target}(x, y, t) = I_{max} \exp[-\frac{1}{2\sigma_g^2}\{[x-x_{peak}(t)]^2 + [y-y_{peak}(t)]^2\}] \tag{2-1}$$

where $(x_{peak}(t), y_{peak}(t))$ locates the center of the pattern relative to the center of the 8-by-8 tracking array, I_{max} is



Circular Equal-intensity Contours

Figure 1

the peak intensity value, and σ_g is the glint dispersion. I_{max} and σ_g were assumed to be known and the apparent target location is actually due to the sum of target dynamics, atmospheric disturbances, and tracker vibration. Since a ground-based stable platform was assumed, vibration effects were ignored and thus

$$\begin{aligned} x_{peak}(t) &= x_d(t) + x_a(t) \\ y_{peak}(t) &= y_d(t) + y_a(t) \end{aligned} \qquad (2\text{-}2)$$

where the subscript d and a represent target dynamics and atmospheric effects, respectively.

An independent first-order Gauss-Markov model in each direction was chosen to represent the benign target motion. Accounting for time-correlated behavior, the model is produced by

$$\dot{x}_d(t) = -(1/T_d)\,x_d(t) + w_{dx}(t) \qquad (2\text{-}3)$$

where T_d is the characteristic correlation time of the target. The additive noise w_{dx} is zero-mean white Gaussian with autocorrelation function

$$E\{w_{dx}(t)w_{dx}(t+\tau)\} = [2\sigma_d{}^2/T_d]\delta(\tau) \qquad (2\text{-}4)$$

where σ_d is the rms value of $x_d\,(t)$. A similar set of relations governs $y_d(t)$. Appropriate values of σ_d and T_d were assumed known.

The atmosphere disturbance terms, $x_a(t)$ and $y_a(t)$, referred to as jitter, have been shown (Refs 1;4) to be adequately modelled as the output of a third-order linear shaping filter with transfer function

$$G(s) = \frac{K\omega_1\omega_2{}^2}{(s+\omega_1)(s+\omega_2)^2} \tag{2-5}$$

driven by unit-strength white Gaussian noise. The values used were ω_1=14 rad/s, ω_2=660 rad/s, and K was adjusted to obtain the desired rms jitter output contribution. Since $\omega_1 << \omega_2$ and the lower frequencies are of importance, the filter approximated the transfer function as $K\omega_1(s+\omega_1)^{-1}$.

The measurement observation was modelled as the intensity pattern of (2-1) corrupted by background noise and inherent FLIR errors. Letting $z_{jk}(t_i)$ denote the measurement available at time t_i of the average intensity over the pixel in the j^{th} row and k^{th} column of the 8-by-8 array, then

$$\begin{aligned} z_{jk}(t_i) = {} & 1/A_p \iint_{\substack{\text{region of} \\ jk^{th}\text{ pixel}}} I_{target}\{\xi_x, \xi_y, t\}\, d\xi_x\, d\xi_y \; + \\ & n_{jk}(t_i) + b_{jk}(t_i) \end{aligned} \tag{2-6}$$

where A_p is the area of one pixel, $n_{jk}(t_i)$ models FLIR noise effects, and $b_{jk}(t_i)$ models the background effects on the jk^{th} pixel. Arraying the 64 scalar equations (2-6) in a single measurement vector yielded the measurement model

$$\underline{z}(t_i) = \underline{h}[\underline{x}(t_i), t_i] + \underline{n}(t_i) + \underline{b}(t_i) \tag{2-7}$$

with $\underline{x} = [x_d, y_d, x_a, y_a]^T$ as the output of a four-state linear dynamics model (reduced from eight states by the removal of $\omega_2{}^2(s+\omega_2)^{-2}$ from (2-5) in each axis direction).

FLIR and background noises, both vectors of dimension 64, were assumed independent of each other so that their variances added for a given pixel. In addition, $\underline{n}(t_i)$ and $\underline{b}(t_i)$ were each assumed to be both spatially and temporally uncorrelated and stationary processes.

A standard extended Kalman filter was generated. With four states and 64 measurements, the inverse covariance form (Ref 7:238-241) of measurement update was employed to avoid the computational burden of large matrix inversions. Also due to computational considerations, the integral in (2-6) was approximated by the integrand evaluated at the center of the appropriate pixel.

The filter equations used for propagating the state estimate, $\hat{\underline{x}}$, and error covariance, $\underline{P}$, from sample time t_{i-1}^+ (after measurement update of t_{i-1}) to time t_i^- (before measurement update) became:

$$\hat{\underline{x}}(t_i^-) = \underline{\Phi}\hat{\underline{x}}(t_{i-1}^+) \tag{2-8}$$

$$\underline{P}(t_i^-) = \underline{\Phi}\underline{P}(t_{i-1}^+)\,\underline{\Phi}^T + \underline{Q}_d \tag{2-9}$$

where the state transition matrix, $\underline{\Phi}$, and input covariance

$$\underline{Q}_d = \int_{t_{i-1}}^{t_i} \underline{\Phi}(t_i - \tau)\underline{Q}\underline{\Phi}^T(t_i - \tau)\,d\tau \tag{2-10}$$

are constant and readily calculated offline. $\underline{Q}$ in (2-10) represents the strength of the continuous time stationary white Gaussian dynamics driving noise.

After combining the effects of $\underline{n}$ and $\underline{b}$ in (2-7) into a single vector $\underline{v}$, with autocorrelation

$$E\{\underline{v}(t_i)\underline{v}^T(t_j)\} = \begin{cases} R\underline{I} & t_i=t_j \\ \underline{0} & t_i \neq t_j \end{cases} \tag{2-11}$$

the measurement update became,

$$\underline{P}^{-1}(t_i^+) = \underline{P}^{-1}(t_i^-) + \underline{H}^T(t_i)\ \underline{R}^{-1}(t_i)\ \underline{H}(t_i) \tag{2-12}$$

$$\underline{P}(t_i^+) = [\underline{P}^{-1}(t_i^+)]^{-1} \tag{2-13}$$

$$\underline{K}(t_i) = \underline{P}(t_i^+)\ \underline{H}^T(t_i)\ \underline{R}^{-1}(t_i) \tag{2-14}$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i)\ \{\underline{z}(t_i) - \underline{h}\ [\hat{\underline{x}}(t_i^-),\ t_i]\} \tag{2-15}$$

where $\underline{H}(t_i)$ is the partial of $\underline{h}$ with respect to $\underline{x}$, evaluated at $\hat{\underline{x}}(t_i^-)$, and $\underline{R}^{-1}(t_i)$ is constant and is generated once offline as $[1/R]\ \underline{I}$ using (2-11).

The tracker thus formulated consistently outperformed a correlation tracker under identical simulated conditions. However, robustness studies conducted by Harnly and Jensen showed the serious problems that arise when the filter did not correctly model true conditions. (Refs 11; 10) With an eye toward generating a tracker capable of tracking air-to-air missiles, the robustness studies yielded insights into additional needed aspects in the tracker design. These were summarized as:

1) ability to estimate size, shape, and orientation of the target image.
2) Online estimation of target intensity height, I_{max}, since it is uncertain, varying, and important to

filter residual generation and tracking performances.

3) Ability to predict future position by maintaining at least a velocity estimate in addition to a position estimate (acceleration estimates may well also be required);
4) Adaptation to maneuvers (detecting a maneuver not predicted by the filter, via residual monitoring, and responding appropriately through gain changing). (Ref 9:p6)

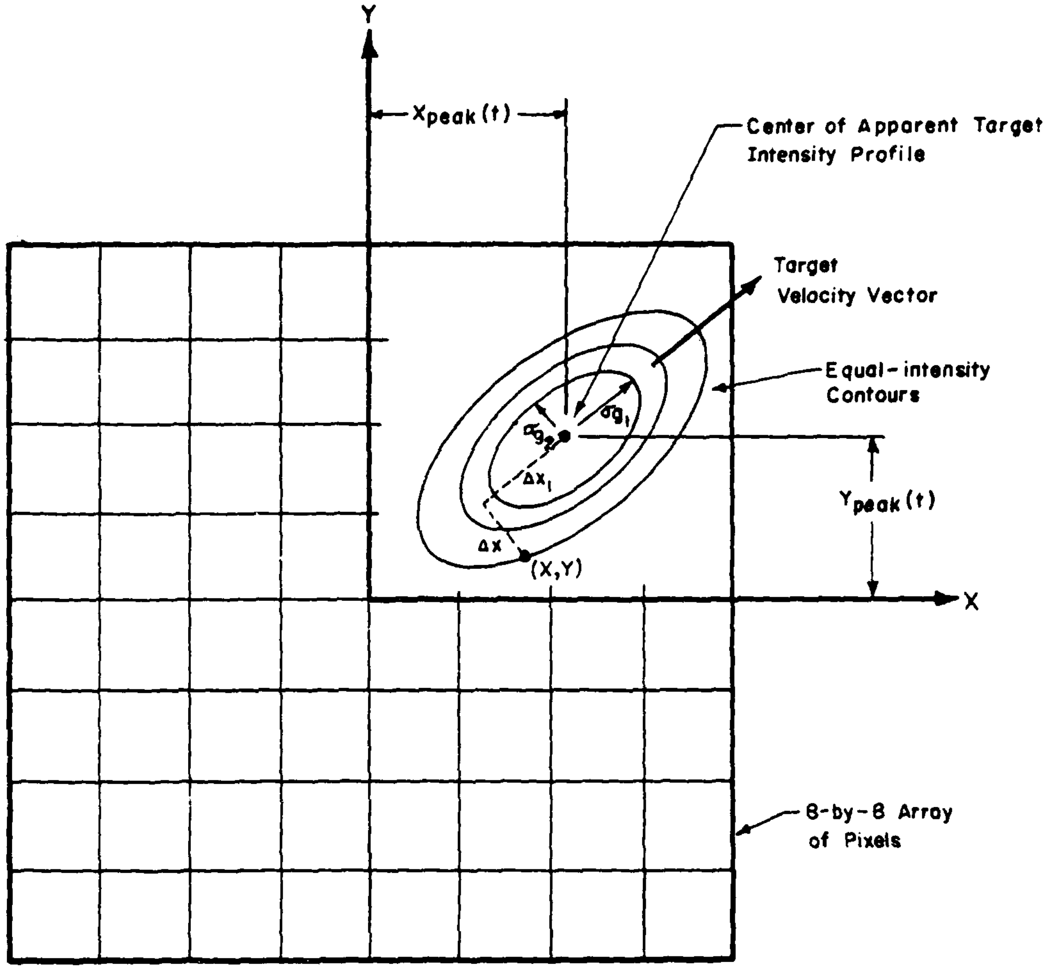
Air-to-Air Missile Tracking Task

Improved Target Intensity Model. The task of tracking the more dynamic air-to-air missile began with an analysis of real FLIR missile image data. The results showed that the image could be well represented by a bivariate Gaussian intensity pattern with elliptical intensity contours as in Fig 2. (Refs 6:p6; 9:p6; 12:p173) By ignoring small angle of attack and sideslip angle, the semimajor axis of the ellipse was aligned with the velocity vector. Letting Δx_1 and Δx_2 be measured from $(x_{peak}(t), y_{peak}(t))$ along the principal axes, the target intensity model (2-1) became

$$I_{target}(\Delta x_1, \Delta x_2, t) = I_{max}\exp\{-\tfrac{1}{2}[(\Delta x_1/\sigma_{g_1})^2 + (\Delta x_2/\sigma_{g_2})^2]\} \tag{2-16}$$

where I_{max}, σ_{g_1}, and σ_{g_2} were treated as uncertain parameters to be estimated simultaneously with the states.

Estimates of σ_{g_1} and σ_{g_2} were generated by a recursive



Elliptical Equal-intensity Contours

Figure 2

gradient algorithm for minimizing the quadratic cost

$$C[\underline{Z}(t_i), \underline{a}] = \{\underline{z}(t_i) - \underline{h}[\hat{\underline{x}}(t_i^-), t_i; \underline{a}]\}^T \{\underline{z}(t_i) - \underline{h}[\hat{\underline{x}}(t_i^-), t_i; \underline{a}]\} \quad (2\text{-}17)$$

as a function of $\underline{a}$, the vector of uncertain parameters. I_{max}, on the other hand, was estimated by a combination of measurement information. The estimate, $\hat{I}$, was calculated by taking the maximum observed pixel intensity value, correcting it with a bias function, and then time averaging with previous estimates. (Ref 6; 9; 12)

Improved Dynamics Model. In order to provide the velocity estimate required for orienting the elliptical contours and to generate appropriate tracking commands for a laser pointing system, a simple six-state filter was implemented. It was based on the dynamics model

$$\begin{aligned} \dot{\underline{x}}(t) &= \underline{v}(t) \\ \dot{\underline{v}}(t) &= \underline{w}(t) \end{aligned} \quad (2\text{-}18)$$

where $\underline{x}(t)$ is target position, $\underline{v}(t)$ is target velocity, and $\underline{w}(t)$ is white Gaussian noise with autocorrelation $E\{\underline{w}(t)\underline{w}^T(t+\tau)\} = \underline{Q}(t)\delta(t)$. $\underline{Q}(t)$ was tuned either online or offline to represent target maneuverability. However, as the tracker rotates to maintain lock on the target, an unmodelled non-inertial acceleration is created on the FLIR image plane.

Desiring to track more dynamic targets, Harnly and Jensen reduced this problem by introducing an eight-state filter which estimated acceleration in the FLIR plan based on the model

$$\dot{\underline{x}}(t) = \underline{v}(t)$$
$$\dot{\underline{v}}(t) = \underline{a}(t)$$
$$\dot{\underline{a}}(t) = \underline{w}(t)$$

where w(t) now models the derivative of the target's acceleration, $\dot{\underline{a}}(t)$. The resulting state vector was

$$\underline{x} = [x_d, y_d, \dot{x}_d, \dot{y}_d, \ddot{x}_d, \ddot{y}_d, x_a, y_a]^T \tag{2-20}$$

Harnly and Jensen concluded their study by looking at adaptive tuning and maneuver indicators. The dynamic noise covariance matrix, $\underline{Q}(t)$, was adaptively tuned using an approximation to maximum likelihood estimation. However, the filter's resulting self tuning ability was not enough to track abrupt harsh maneuvers. Such maneuvers required development of maneuver detection indicators which signaled a need for additional responses within the filter. Harnly and Jensen incorporated two responses, (1) increasing the gain by reverting to an acquisition cycle of very high reinitialized $\underline{P}$ and high $\underline{Q}_d$, and (2) using updated state estimates for reprocessing the previous sample period's state estimate time propagation and for future propagation, for use during an ensuing 0.5 second reacquisition cycle.

Baseline Extended Kalman Filter

The starting point for the current study was an extended Kalman filter (EKF) based on the Harnly and Jensen eight-state filter with elliptical equal-intensity contours in the target intensity model. Considering the scope of this study, only the adaptive size and shape estimation through parameters σ_{g_1} and σ_{g_2} was maintained, thus providing a straightforward filter as the standard for comparison studies. Offline tuning of I_{max} and $\underline{Q}(t)$ replaced the adaptive tuning in order to provide flexibility in establishing best performance standards. Further, the rather ad hoc maneuver indicators and associated filter responses were deleted, so as not to cloud the performance comparisons in this study. Performance evaluation of this filter served as the standard for comparative evaluation with the alternative filters pursued in the next section.

The detailed equations for this filter are presented in Appendix A, and the associated computer software is contained in Appendix D.

III. Alternative Filters

Inherent in the formulation of the extended Kalman filter in Section II is the assumption that the nonlinear target intensity function (2-6) can be linearized about $\hat{\underline{x}}(t_i^-)$ using a first order Taylor Series approximation. This linearization process assumes that the process errors which accumulate between measurement updates remain small. One method of keeping the error accumulation small is to decrease the time between updates. (Ref 2:p24) However, in this problem mechanization of the FLIR sensor fixes the sample rate. Thus, other approaches must be considered since, as Harnly and Jensen reported, the linearization breaks down for highly maneuvering targets.

As a result, this study looks at several alternative filters in hopes of gaining improved tracking performance. The first of these, the extended Kalman Filter with bias correction term, is derived from a Gaussian second order filter. Its additional second order information often helps the EKF, which neglected higher than first order terms. A constant gain EKF and a constant gain EKF with bias correction term were then evaluated. Often, they demonstrate better robustness properties than the standard EKF. Finally, as an alternative to these two conditional moment estimators, a statistically linearized estimator is developed. Statistics within the filter are used to linearize the process using a power series representation.

Besides tracking performance, computational burden is always an issue due to online computer hardware limitations. This also was an important consideration for pursuing the constant gain filters.

Extended Kalman Filter with Bias Correction Term

Approximate conditional moment extimators, which includes the linear Kalman Filter and linearized extended Kalman filter, provide an estimate which is an approximate evaluation of the conditional mean

$$\underline{\hat{x}}(t_i/t_{i-1}) \triangleq E\{\underset{\sim}{\underline{x}}(t_i) | \underset{\sim}{\underline{Z}}(t_{i-1}) = \underline{Z}_{i-1}\} \tag{3-1a}$$

$$\underline{\hat{x}}(t_i/t_i) \triangleq E\{\underset{\sim}{\underline{x}}(t_i) | \underset{\sim}{\underline{Z}}(t_i) = \underline{Z}_i\} \tag{3-1b}$$

where (3-1a) is the estimate after propagating from time t_{i-1} to t_i and (3-1b) is the estimate after incorporating the measurement at time t_i. These two values are also referred to as $\underline{\hat{x}}(t_i^-)$ and $\underline{\hat{x}}(t_i^+)$ respectively. Exact evaluation of these conditional expectations requires knowledge of the entire conditional density functions

$$f_{\underset{\sim}{\underline{x}}(t_i) | \underset{\sim}{\underline{Z}}(t_{i-1})}(\underline{\xi} | \underline{Z}_{i-1}) \tag{3-2a}$$

$$f_{\underset{\sim}{\underline{x}}(t_i) | \underset{\sim}{\underline{Z}}(t_i)}(\underline{\xi} | \underline{Z}_i) \tag{3-2b}$$

which includes all higher moments as well as the conditional mean and covariance. (Ref 8:Chapt 12) In the Kalman filter for a problem defined by linear dynamic and measurement models, driven by white Guassian noise, the first two moments completely describe the density function which is Gaussian. The resulting conditional mean is then the optimal estimate from a Bayesian viewpoint. (Ref 7:p215) However, for nonlinear processes, the optimal estimator will generally be infinite dimensional. The Gaussian second order filter assumes that the conditional density is nearly Gaussian, so that third and higher odd central moments are essentially zero, and that fourth and higher even central moments are expressible in terms of the covariance. In this study, sixth and higher even moments are also neglected. This yields an algorithm that does not require evaluation of an infinite number of moments.

Since a linear dynamics model was assumed for this tracker, attention was confined to the nonlinear measurement update. Based on work by Kramer and Jazwinski, one approximation technique for evaluating (3-2) is presented in (Ref 8:Chapt 12). This approach assumes that the mean and covariance after update are expressible as a power series in the residual,

$$\{\underline{z}(t_i) - \hat{\underline{z}}(t_i^-)\} \overset{\Delta}{=} \{\underline{z}(t_i) - E[\underline{z}(t_i) | \underline{z}(t_{i-1}) = \underline{z}(t_{i-1},\cdot)]\} \tag{3-3}$$

which is truncated at first order terms:

$$\hat{\underline{x}}(t_i^+) = \underline{a}_0 + \underline{a}_1\{\underline{z}(t_i) - \hat{\underline{z}}(t_i^-)\} \tag{3-4a}$$

$$\underline{P}(t_i^+) = \underline{b}_0 + \underline{b}_1\{\underline{z}(t_i) - \hat{\underline{z}}(t_i^-)\} \tag{3-4b}$$

The coefficients in (3-4) are then evaluated using Bayes Rule and truncated Taylor Series representations. This evaluation is described in detail in (Ref 8:Chapt 12). The resulting

Gaussian second order (GS) filter is further modified by setting $\underline{b}_1$ in (3-4b) to zero. This eliminates a random residual forcing function which allowed negative computed values of $\underline{P}(t_i^+)$. The resulting measurement update equations are:

$$\underline{A}_{GS}(t_i) = \underline{H}[\hat{\underline{x}}(t_i^-), t_i]\, \underline{P}(t_i^-)\, \underline{H}^T[\hat{\underline{x}}(t_i^-), t_i] + \hat{\underline{B}}_m(t_i^-) + \underline{R}(t_i) \qquad (3\text{-}5)$$

$$\underline{K}_{GS}(t_i) = \underline{P}(t_i^-)\underline{H}^T\, [\hat{\underline{x}}(t_i^-), t_i]\, \underline{A}_{GS}^{-1}\, (t_i) \qquad (3\text{-}6)$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}_{GS}(t_i)\{\underline{z}_i - \underline{h}[\hat{\underline{x}}(t_i^-), t_i] - \hat{\underline{b}}_m(t_i^-)\} \qquad (3\text{-}7)$$

$$\underline{P}(t_i^+) = \underline{P}(t_i^-) - \underline{K}_{GS}(t_i)\underline{H}[\hat{\underline{x}}(t_i^-), t_i]\underline{P}(t_i^-) \qquad (3\text{-}8)$$

with the defining relations

$$\underline{H}[\hat{\underline{x}}(t_i^-), t_i] \overset{\Delta}{=} \frac{\partial \underline{h}[\underline{x}, t_i]}{\partial \underline{x}} \bigg|_{\underline{x}=\hat{\underline{x}}(t_i^-)} \qquad (3\text{-}9)$$

$$\hat{\underline{B}}_{m_{k\ell}}{}^{th}\text{element} \overset{\Delta}{=} \tfrac{1}{2}\mathrm{tr}\Big\{\frac{\partial^2 h_k[\underline{x}, t_i]}{\partial \underline{x}^2}\bigg|_{\underline{x}=\hat{\underline{x}}(t_i^-)} \underline{P}(t_i^-) \cdot \frac{\partial^2 h_\ell \underline{x}[(t_i), t_i]}{\partial \underline{x}^2}\bigg|_{\underline{x}=\hat{\underline{x}}(t_i^-)} P(t_i^-)\Big\} \qquad (3\text{-}10)$$

$$\underline{b}_{m_k}{}^{th}\text{element} \overset{\Delta}{=} \tfrac{1}{2}\mathrm{tr}\Big\{\frac{\partial^2 h_k[\underline{x}, t_i]}{\partial \underline{x}^2}\bigg|_{\underline{x}=\hat{\underline{x}}(t_i^-)} \underline{P}(t_i^-)\Big\} \qquad (3\text{-}11)$$

where $h_k[\underline{x}, t_i]$ is the k^{th} element of $\underline{h}[\underline{x}, t_i]$.

The structure of this algorithm is similar to that of the EKF in Section II. Note however, that this form requires additional burdensome second moment calculations. Also new to the structure is the vector $\hat{\underline{b}}_m(t_i)$ in the residual generation. A first order EKF often generates biased estimates due to neglected nonlinearities, but the addition of the second order information contained in $\hat{\underline{b}}_m(t_i)$ reduces the bias, hence $\hat{\underline{b}}_m(t_i)$ is the "bias correction" term. (Ref 3:p507)

In order to make use of second order information without incurring the computational burden of the full-scale Gaussian second order filter, just the bias correction term is added to a first order EKF. Specifically, the residuals are generated as in (3-7), but the filter gain and covariance are computed as in the EKF. The performance improvement from such a filter over that of an EKF is related to the harshness of the nonlinearity. Thus, substantial enhancement is expected when large second partial derivative values develop.

As implemented for this study, the baseline EKF was revised to include the additional bias correction terms. The second partial derivatives were computed in subroutine MEASF (see Appendix B and D) at the same time as the first partial derivatives. An additional subroutine, BIASCT, performed the matrix multiplication and trace operation needed in (3-11).

Constant Gain Filters

For a time-invariant nonlinear process acting along a constant nominal state trajectory, the linear perturbation equations are time invariant. The gain, $\underline{K}(t_i)$, associated with the linearized filter for such a trajectory can reach a steady state value, $\underline{K}_{ss}$. (Ref 8:Chapt 9) Then the constant-gain EKF for this study is defined by

$$\hat{\underline{x}}(t_i) = \underline{\Phi}\hat{\underline{x}}(t_i^{+}) \tag{3-12}$$

$$\hat{\underline{x}}(t_i^{+}) = \hat{\underline{x}}(t_i^{-}) + \underline{K}_{ss}\{\underline{z}_i - \underline{h}[\hat{\underline{x}}(t_i^{-})]\} \tag{3-13}$$

Likewise, the constant-gain EKF with bias correction terms is defined by (3-12) using the steady state $\underline{P}(t_i^{-})$ and

$$\hat{\underline{x}}(t_i^{+}) = \hat{\underline{x}}(t_i^{-}) + \underline{K}_{ss}\{\underline{z}_i - \underline{h}[\hat{\underline{x}}(t_i^{-})] - \hat{\underline{b}}_m(t_i^{-})\} \tag{3-14}$$

Notice that these filter forms do not require knowledge of the conditional covariance, $\underline{P}(t_i^{-})$, thus significantly reducing computational burden. In addition, they may also demonstrate good robustness characteristics since their gain calculation is not effected by filter mismodelling. (Ref 19:p721)

This study used a simple procedure for obtaining $\underline{K}_{ss}$. First, the filter-computed gain history, $\underline{K}(t_i)$, was recorded during Monte Carlo simulation runs. Then, a $\underline{K}_{ss}$ was chosen based on these values in hope of gaining good performance for both the 5g and 10g trajectories described in the next section (see Appendix C).

An alternative method for obtaining $\underline{K}_{ss}$ is applying pole placement or more extensive eigenvalue/eigenvector assignment techniques to the linearized perturbation equations. This method was not pursued in this study.

Statistically Linearized Filter

In contrast to dealing with the nonlinear measurement function $\underline{h}[\underset{\sim}{x}(t_i),t_i]$ in (2-6) by linearizing with a first order Taylor series approximation, the statistically linearized (SL) filter approximates $\underline{h}[\underset{\sim}{x}(t_i),t_i]$ with the first two terms in a power series

$$\underline{h}[\underset{\sim}{x}(t_i),t_i] \cong \underline{h}_0(t_i) + \underline{H}(t_i)\underset{\sim}{x}(t_i) \tag{3-15}$$

with $\underline{h}_0(t_i)$ and $\underline{H}(t_i)$ determined through an optimization technique. As a result, $\underline{h}[\underset{\sim}{x}(t_i), t_i]$ need not be differentiable with respect to $\underline{x}(t_i)$ as required for the Taylor series representation. However, determining the coefficients $\underline{h}_0(t_i)$ and $\underline{H}(t_i)$ requires the evaluation of several conditional expectations. Without the assumptions made in the following development, this technique could become infinite dimensional.

The representation in (3-15) can be optimized with respect to the error variable

$$\underset{\sim}{e}(t) = \underline{h}[\underset{\sim}{x}(t_i), t_i] - [\underline{h}_0(t_i) + \underline{H}(t_i)\underset{\sim}{x}(t_i)] \tag{3-16}$$

so that the assumed form has minimum mean square error, i.e. minimum value of:

$$J = E\{\underset{\sim}{e}^T(t)\underline{W}\underset{\sim}{e}(t)|\underset{\sim}{Z}(t_i) = \underline{Z}_i\} \tag{3-17}$$

where $\underline{W}$ is a general weighting matrix. (Ref 8:Chapt 12) Evaluating the expectation in (3-17) requires knowledge of the conditional density function (3-2b). Conducting the optimization yields the coefficients

$$\underline{h}_0(t_i) = \widehat{\underline{h}[\underset{\sim}{x}(t_i),t_i]} - \underline{H}(t_i)\hat{\underline{x}}(t_i^-) \tag{3-18}$$

$$\underline{H}(t_i) = \{\widehat{\underline{h}[\underset{\sim}{x}(t_i),t_i]\underset{\sim}{x}^T(t_i)} - \widehat{\underline{h}[\underset{\sim}{x}(t_i),t_i]}\hat{\underline{x}}(t_i^-)\}\cdot \underline{P}^{-1}(t_i^-) \tag{3-19}$$

where

$$(\cdot) \triangleq E\{(\cdot)|\underset{\sim}{Z}(t_{i-1}) = \underline{Z}_{i-1}\} \tag{3-20}$$

$$\underline{P}(t_i^-) = \widehat{[\underset{\sim}{x}(t_i)-\hat{\underline{x}}(t_i^-)][\underset{\sim}{x}(t_i)-\hat{\underline{x}}(t_i^-)]^T} \tag{3-21}$$

This approximation then yields measurement update equations at time t_i of

$$\underline{K}_{SL}(t_i) = \underline{P}(t_i^-)\underline{H}^T(t_i)[\underline{H}(t_i)\underline{P}(t_i^-)\underline{H}^T(t_i)+\underline{R}(t_i)]^{-1} \tag{3-22}$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-)+\underline{K}_{SL}(t_i)\{\underline{z}(t_i)-\widehat{\underline{h}[\underset{\sim}{x}(t_i),\ t_i]}\} \tag{3-23}$$

$$\underline{P}(t_i^+) = \underline{P}(t_i^-)-\underline{K}_{SL}(t_i)\underline{H}(t_i)\underline{P}(t_i^-) \tag{3-24}$$

with $\underline{H}(t_i)$ given by (3-19). The conditional expectations $\widehat{\underline{h}\,\underline{x}^T}$, $\hat{\underline{h}}$, and $\hat{\underline{x}}$ in the above relationships are computed by assuming that the conditional density function (3-2b) is Gaussian with mean $\hat{\underline{x}}(t_i^-)$ and covariance $\underline{P}(t_i^-)$, as generated by the previous propagation.

Notice the structural similarity of (3-22, 23, 24) to the EKF update equations (2-12, 13, 14, 15), where now the statistically optimized $\underline{H}(t_i)$ replaces the matrix of partial derivatives, $\underline{H}(t_i^-)$. This structure accounts for variations in $\underset{\sim}{x}(t_i)$ away from $\hat{\underline{x}}(t_i^-)$, which invalidate first order Taylor series representations. As a result, better performance is expected, especially for cases which have large error covariance magnitudes.

Evaluating $\widehat{\underline{h}[\underline{x}(t_i),t_i]}$. As noted before, this filter requires evaluating several conditional expectations. Look first at evaluating the term $\hat{h}_k$ for k = 1, 2, 3,. . ., 64

$$\widehat{h_k[\underline{x}(t_i),t_i]} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\Big[1/A_p \iint_{\substack{\text{region of}\\ jk^{th}\text{ pixel}}} I_{target}[\Delta x_1, \Delta x_2, t_i]\,dx\,dy\Big] f_{(x_{peak},y_{peak})|\underline{Z}}(\xi_x,\xi_y,|\underline{Z}_i)\,d\xi_x d\xi_y \tag{3-25}$$

where I_{target} is defined in (2-16). The approximate numerical evaluation of (3-25) evolved from a physical interpretation. First, $\underline{h}[\underline{x}(t_i),t_i]$ takes on different values for each realizations of $\underline{x}(t_i)$. Evaluating $\underline{h}[\underline{x}(t_i), t_i]$ however, requires only the position state information contained in $x_{peak}(t_i)$ and $y_{peak}(t_i)$. Thus for each ℓ^{th} realization of $\underline{x}(t_i)$, $\underline{x}_\ell(t_i)$, $[x_{peak}(t_i), y_{peak}(t_i)]_\ell$ is formed and $h[x_{peak}(t_i), y_{peak}(t_i), t_i]_\ell$ is then evaluated. Second, the conditional probability density function, assumed Gaussian, is generated and integrated to yield

$$P_\ell[\hat{x}_{peak}(t_i^-), \hat{y}_{peak}(t_i^-), {}^{\sigma}x_{peak}, {}^{\sigma}y_{peak}] = \iint_{A_\ell} f_{(x_{peak},y_{peak})|\underline{Z}}(\xi_x,\xi_y|\ \underline{Z}_{i-1})\,d\xi_x d\xi_y \tag{3-26}$$

where A_ℓ is an infinitesimally small region surrounding $[x_{peak}(t_i), y_{peak}(t_i)]_\ell$. P_ℓ is then the appropriate probability that $[x_{peak}(t_i), y_{peak}(t_i)]_\ell$ locates the peak of the true apparent target intensity function. The final step in this interpretation moves through realizations $\underline{x}_\ell(t_i)$, evaluating

$\underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_\ell$ and P_ℓ. Multiplying these two values together for each realization and then summing over an inifinite number of $\underline{x}_\ell(t_i)$ values approximates (3-25).

$$\widehat{\underline{h}[\underline{x}(t_i), t_i]} \cong \sum_{\ell=1}^{\infty} P_\ell \cdot \underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_\ell \qquad (3\text{-}27)$$

However, implementing such a summation is impractical and so a discrete approximation is pursued. The approximation will seek to use the finite pixel area as an appropriate region A_ℓ in order to exploit the EKF measurement software based on one pixel discretizations. This three step approach led to the algorithm in Appendix C, whose detailed development follows.

The first step, evaluating $\underline{h}[\underline{x}(t_i), t_i]$ at each $[x_{peak}(t_i), y_{peak}(t_i)]_\ell$, uses the same software which evaluates $\underline{h}[\hat{\underline{x}}(t_i), t_i]$ in the EKF. As before, the value of the intensity function (2-16) at the pixel's center approximates the average over the entire pixel.

The second step, evaluating P_ℓ, follows a different approach. Generating the Gaussian conditional density function imbedded in (3-2a), $f_{(x_{peak}, y_{peak})|\underline{Z}}$, first requires the conditional covariance and mean, $\underline{P}(t_i)$ and $\hat{\underline{x}}(t_i^-)$. Using the definition of $x_{peak}(t)$ and $y_{peak}(t)$ in (2-2) and the state vector (2-20) transforms the desired density function into the FLIR x-y coordiante frame.

$$\begin{aligned} x_{peak}(t) &= x_d(t) + x_a(t) \\ &= [1\ 0\ 0\ 0\ 0\ 0\ 1\ 0]\ \underline{x}(t) \qquad (3\text{-}28a) \\ &= [T_x]\ \underline{x}(t) \end{aligned}$$

$$\begin{aligned} y_{peak}(t) &= y_d(t) + y_a(t) \\ &= [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1]\underline{x}(t) \qquad (3\text{-}28b) \\ &= [T_y]\underline{x}(t) \end{aligned}$$

The combination of states in (3-28) also combines the atmospheric and dynamic terms of $\underline{P}(t_i^-)$ into separate, single sigma values for both the x and y directions.

$$\sigma^2_{x_{peak}} = [T_x]\underline{P}(t_i^-)[T_x]^T \qquad (3\text{-}29a)$$

$$\sigma^2_{y_{peak}} = [T_y]\underline{P}(t_i^-)[T_y]^T \qquad (3\text{-}29b)$$

The resulting bivariate Gaussian distribution centered at $[\hat{x}_{peak}(t_i^-), \hat{y}_{peak}(t_i^-)]$ simplifies by assuming the stochastic processes associated with the FLIR x and y directions to be uncorrelated. The covariance matrix associated with $f_{(x_{peak}, y_{peak})|\underline{\underline{Z}}}$ then becomes

$$\begin{bmatrix} \sigma^2_{x_{peak}} & 0 \\ 0 & \sigma^2_{y_{peak}} \end{bmatrix} \qquad (3\text{-}30)$$

This allows the evaluation of $f_{(x_{peak}, y_{peak})|\underline{\underline{Z}}}$ to be computed as the product of the two one-dimensional Guassian distributions

$$f_{x|\underline{\underline{Z}}}(\xi|\underline{Z}_{i-1}) = [1/2\pi\sigma_{x_{peak}}]\exp\{-\tfrac{1}{2}[\xi - \hat{x}_{peak}(t_i^-)]^2 \cdot 1/\sigma^2_{x_{peak}}\} \qquad (3\text{-}31a)$$

$$f_{y|\underset{\sim}{\underline{Z}}}(\rho|\underline{Z}_{i-1}) = [1/2\pi\sigma_{y_{peak}}]\exp\{-\tfrac{1}{2}[\rho-\hat{y}_{peak}(t_i^-)]^2 \cdot 1/\sigma^2_{y_{peak}}\} \tag{3-31b}$$

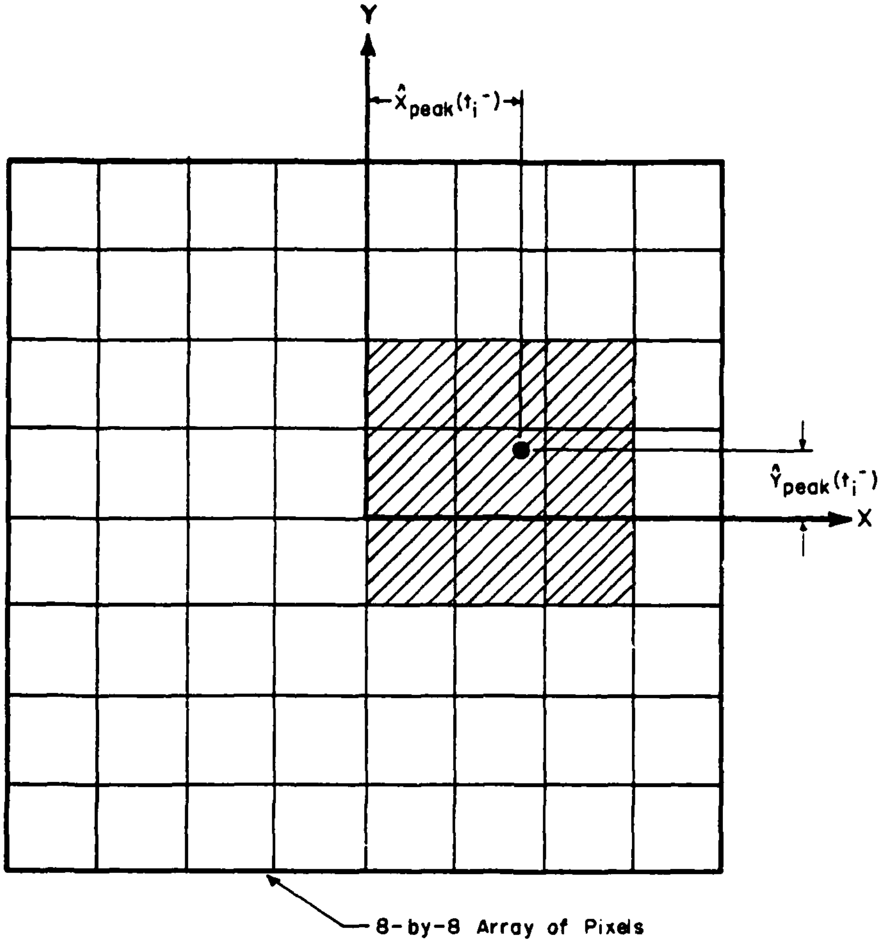
Note that this assumption can be verified by computing the off-diagonal terms in (3-30) as $[T_x]\underline{P}(t_i^-)[T_y]^T$. Consistently, the diagonal terms in (3-30) are an order of magnitude larger than the off-diagonal correlation terms, which justifies such as assumption.

A second assumption further eases the computational burden of evaluating $f_{(x_{peak},y_{peak})|\underline{Z}}$. Examining the sigma values, $\sigma_{y_{peak}}$ and $\sigma_{x_{peak}}$, in the results from the other filters showed how large a region had significant probability of containing $[x_{peak}(t_i),y_{peak}(t_i)]$. This examination found typical 3σ values of 0.97 pixels in the x direction and 0.72 pixels in the y direction. Thus, the evaluation of P_ℓ is limited to the pixel containing $[x_{peak}(t_i^-),y_{peak}(t_i^-)]$ and its eight nearest neighbor pixels as shown in Fig 3.

The third and final step in evaluating (3-25) is an implementation of the summation

$$\widehat{\underline{h}[\underset{\sim}{\underline{x}}(t_i),t_i]} \cong \sum_{\ell=1}^{9} P_\ell(t_i) \cdot \underline{h}[x_{peak}(t_i),y_{peak}(t_i),t_i]_\ell \tag{3-32}$$

where the summation index ℓ represents the nine pixels of interest. The number of realizations of $[x_{peak}(t_i),y_{peak}(t_i)]_\ell$ is limited to nine in consideration of computational burden. In line with the approximately zero-mean error assumption,



Region for Evaluating P_ℓ

Figure 3

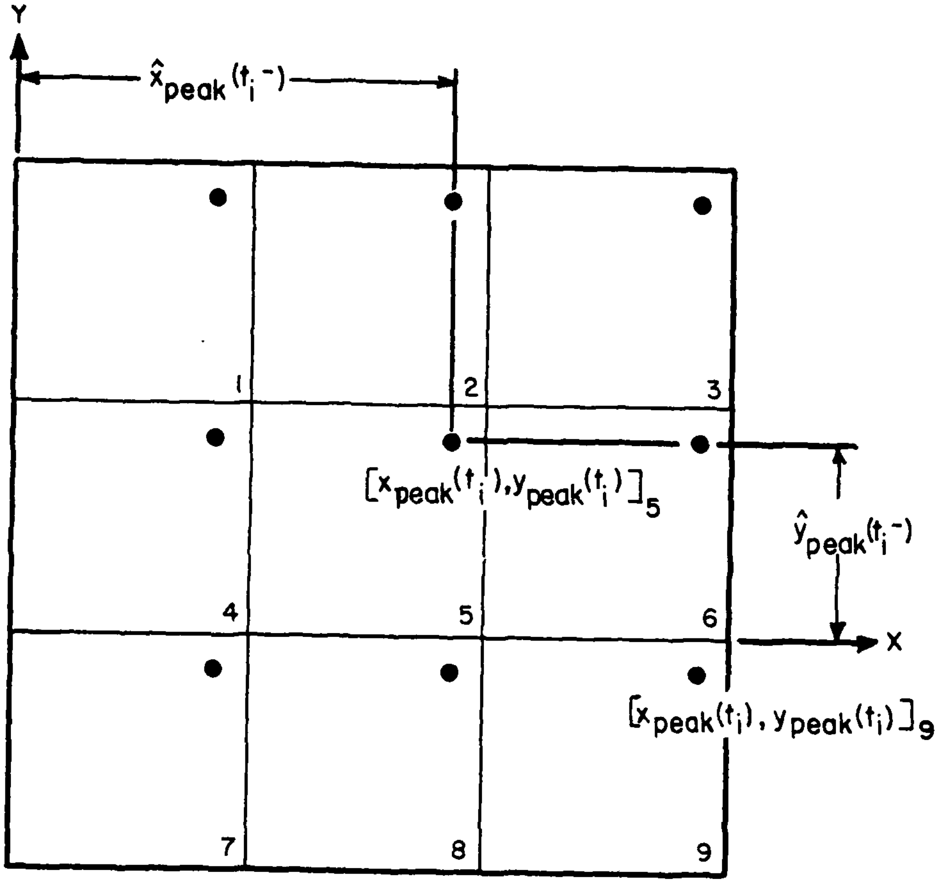
this study used for its nine points $[\hat{x}_{peak}(t_i^-), \hat{y}_{peak}(t_i^-)]$ and one point in each of the neighboring pixels corresponding to a one pixel shift in the x and/or y directions from $[\hat{x}_{peak}(t_i^-), \hat{y}_{peak}(t_i^-)]$. These nine realizations are shown in Fig 4. Thus an appropriate region A_ℓ in (3-2a) is the pixel itself. Fig 5 summarizes the evaluation of the P_ℓ values. Integrations are approximated by evaluating the function in (3-31) at the pixel centers, and as a final step, the individual pixel probabilities P_ℓ are normalized so that the conditional probability density function has unit volume over the nine pixel area. A final computational consideration for (3-32) evaluates $\underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_\ell$ only once for a 10-by-10 pixel array centered about $[\hat{x}_{peak}(t_i^-), \hat{y}_{peak}(t_i^-)]$ at each time t_i. The additional rows and columns are needed to evaluate the terms in (3-32) associated with the eight nearest neighbor pixels. The appropriate 8-by-8 array values for $\underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_\ell$ are then accessed for each iteration in (3-32) as shown in Fig 6.

<u>Evaluating $\widehat{\underline{h}[\underset{\sim}{x}(t_i), t_i]\underset{\sim}{x}^T(t_i)}$.</u> Consider next the conditional expectation

$$\widehat{\underline{h}[\underset{\sim}{x}(t_i), t_i]\underset{\sim}{x}^T(t_i)} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \underline{h}[\underset{\sim}{x}(t_i), t_i]\underset{\sim}{x}^T(t_i) \, f_{(x_{peak}, y_{peak})|\underline{Z}}(\xi_x, \xi_y | \underline{Z}_i) \, d\xi_x d\xi_y \tag{3-33}$$

Intuitively, a summation similar to (3-32) seems appropriate

$$\widehat{\underline{h}[\underset{\sim}{x}(t_i), t_i]\underset{\sim}{x}^T(t_i)} \cong \sum_{\ell=1}^{9} P_\ell(t_i) \cdot \underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_\ell \underline{x}_\ell^T(t_i) \tag{3-34}$$


Y
$\hat{x}_{peak}(t_i^-)$
1
2
3
4
5
6
7
8
9
$[x_{peak}(t_i), y_{peak}(t_i)]_5$
$\hat{y}_{peak}(t_i^-)$
X
$[x_{peak}(t_i), y_{peak}(t_i)]_9$

Realizations of $[x_{peak}(t_i), y_{peak}(t_i)]_\ell$
Figure 4

$$P_1 = P_{x1} P_{y1} / P_T$$

$$P_2 = P_{x2} P_{y1} / P_T$$

$$P_3 = P_{x3} P_{y1} / P_T$$

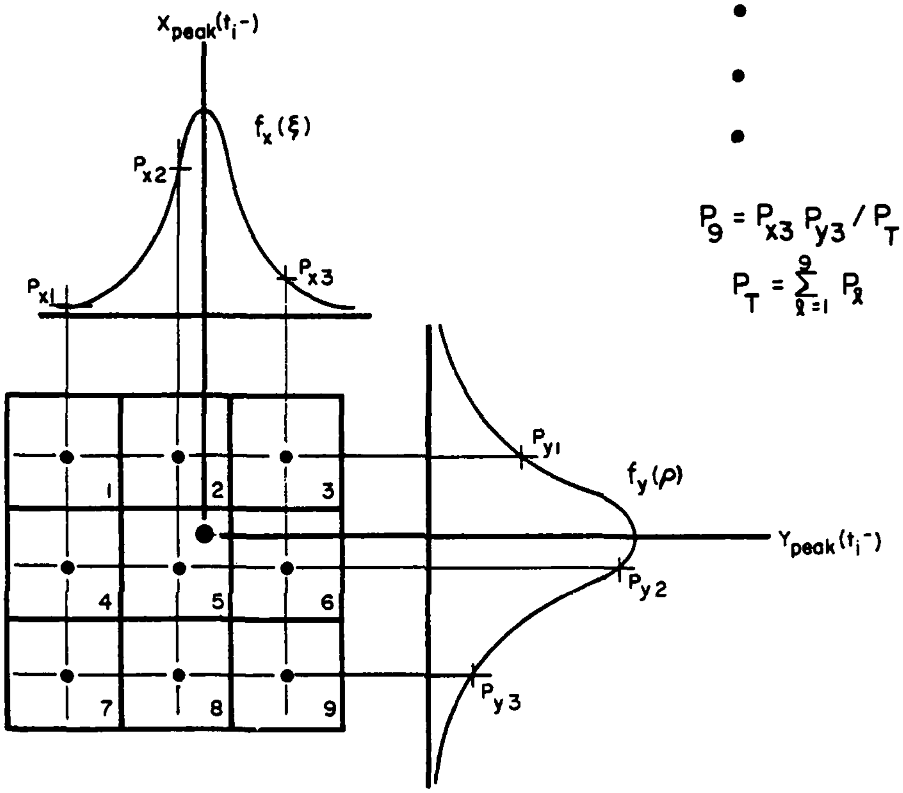
•

•

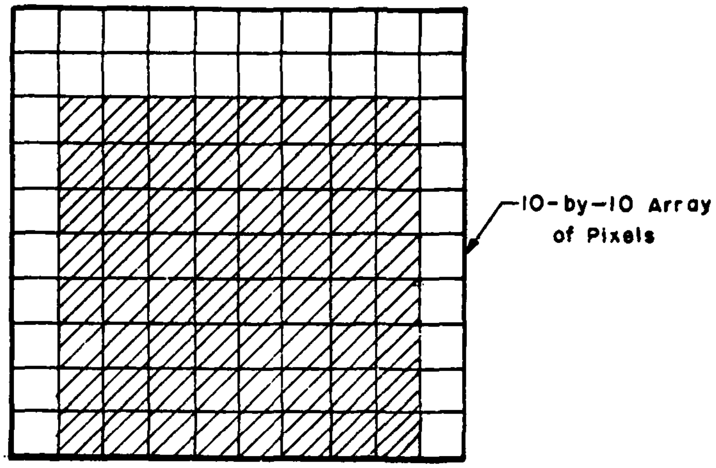
•

$$P_9 = P_{x3} P_{y3} / P_T$$

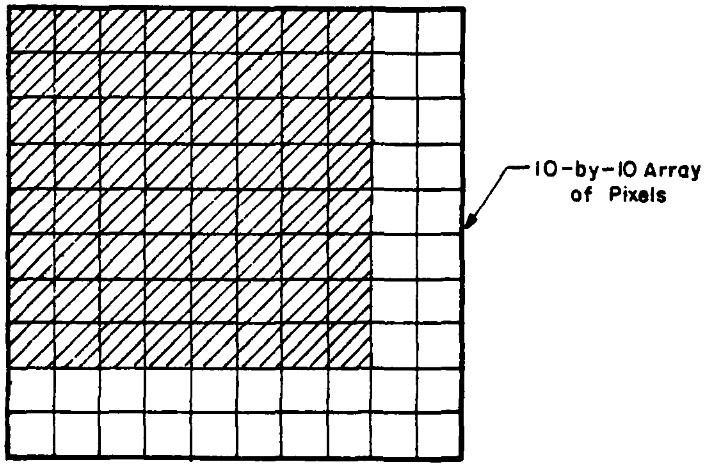
$$P_T = \sum_{\ell=1}^{9} P_\ell$$



Evaluating P_ℓ
Figure 5



VALUES ACCESSED FOR
$\underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_2$



VALUES ACCESSED FOR
$\underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_9$

Access of Values for $\underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_\ell$
Figure 6

where P_ℓ and $\underline{h}[x_{peak}(t_i), y_{peak}(t_i), t_i]_\ell$ are calculated as in the previous section. The realizations of the state vector, $\underline{x}_\ell^T(t_i)$, are based on the points $[x_{peak}(t_i),\ y_{peak}(t_i)]_\ell$. However, each unit pixel shift of either $x_{peak}(t_i)$ or $y_{peak}(t_i)$ is a combination of the atmospheric and dynamic states through the transformation (3-28). Formulating $\underline{x}_\ell^T(t_i)$ requires an indication of how much of the unit pixel shift is caused by dynamics and by atmospheric jitter. Such indicators are the values

$$\delta x_d = \frac{\sigma_{x_d}}{\sigma_{x_d} + \sigma_{x_a}} \qquad \delta y_d = \frac{\sigma_{y_d}}{\sigma_{y_d} + \sigma_{y_a}}$$

$$\delta x_a = \frac{\sigma_{x_a}}{\sigma_{x_d} + \sigma_{x_a}} \qquad \delta y_a = \frac{\sigma_{y_a}}{\sigma_{y_d} + \sigma_{y_a}} \qquad (3\text{-}35)$$

where the sigma values are the square roots of the appropriate diagonal terms of $P(t_i)$, and δx_d and δx_a or δy_d and δy_a are assumed of the same sign. As an example, $\underline{x}_9^T(t_i)$ associated with $[x_{peak}(t_i), y_{peak}(t_i)]_9$ in Fig 4 would be

$$\underline{x}_9^T(t_i) = \hat{\underline{x}}^T(t_i^-) + [\delta x_d \ \ -\delta y_d \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ \delta x_a \ -\delta y_a] \qquad (3\text{-}36)$$

Note that pixel shifts up and shifts right are considered positive.

Algorithm Summary. This study implements the statistically linearized filter by integrating the coefficient $\underline{H}(t_i)$ and predicted measurement $\widehat{\underline{h}[\underset{\sim}{\underline{x}}(t_i), t_i]}$ into the existing EKF software, replacing $\underline{H}(t_i)$ and $\underline{h}[\hat{\underline{x}}(t_i^-),t_i]$, respectively. Subroutine STATLN in Appendix C computes the necessary conditional expectations and $\underline{H}(t_i)$ based on the propagated state, $\hat{\underline{x}}(t_i^-)$, and conditional covariance, $\underline{P}(t_i^-)$. An extension to subroutine MEASF provides the 10-by-10 measurement array as described previously.

As a summary of the statistically linearized filter, the sequence of steps within STATLN follows:

1) locate pixel containing $[\hat{x}_{peak}(t_i^-),\hat{y}_{peak}(t_i^-)]$

2) calculate $^{\sigma}x_{peak}$ and $^{\sigma}y_{peak}$ from $\underline{P}(t_i^-)$

3) evaluate P_ℓ ℓ= 1, 2, 3, ..., 9

4) evaluate $\widehat{\underline{h}[\underset{\sim}{x}(t_i),t_i]}$ as $\sum_{\ell=1}^{9} P_\ell \cdot \underline{h}[x_{peak}(t_i), y_{peak}(t_i),t_i]_\ell$

5) calculate δx_a, δx_d, δy_a, and δy_d from $\underline{P}(t_i^-)$

6) evaluate $\widehat{\underline{h}[\underset{\sim}{x}(t_i),t_i]\underset{\sim}{\underline{x}}^T(t_i)}$ as $\sum_{\ell=1}^{9} P_\ell \cdot \underline{h}[x_{peak}(t_i), y_{peak}(t_i),t_i]_\ell x_\ell^T(t_i)$

7) evaluate $\underline{H}(t_i)$ using equation (3-19)

Equations (3-22) through (3-24 define the measurement update, but as in the EKF it is implemented in the inverse covariance form.

IV. Analysis Method

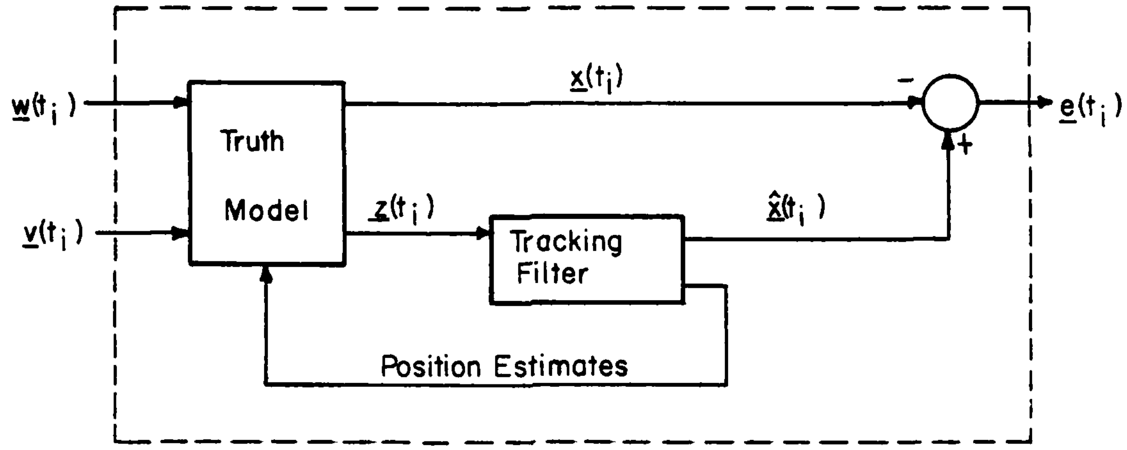
The primary goal of analyzing the filters in this study is to evaluate their relative performance (i.e. the filter's ability to track targets in varying conditions). This is accomplished through a Monte Carlo study. When comparing several filters, however, an analysis should also address the costs associated with obtaining desired performance levels, so computational burden is also evaluated.

Monte Carlo Study

Evaluating a filter's performance requires a description of the estimation errors committed by the filter when subjected to a "real world" environment. Depicting the "real world" with a truth model provides an analysis tool for determining accurate statistics which describe the filter's errors. Two techniques are available to meet this objective, a covariance analysis and a Monte Carlo study. However, the measurement model in this study is nonlinear and nonlinear effects cannot be fully evaluated by a covariance analysis. (Ref 7:p329) This study therefore uses a Monte Carlo technique for evaluating filter performance.

For this study, the "real world" is depicted by a truth model that was first developed by Mercier and then expanded by Harnly and Jensen. (Refs 13; 6) As indicative of the "real world", this model provides for (1) realistic missile dynamics, (2) a full implementation of the third order atmospheric effects as defined in (2-5), and (3) simulated target FLIR and background noises of variable and realistic spatial and temporal correlation. The block diagram in Fig 7 depicts the structure of a Monte Carlo study. Numerically generated white Gaussian noises, $\underline{w}(t_i)$ and $\underline{v}(t_i)$, drive the truth model which produces both sample-by-sample simulations, $\underline{x}(t_i)$, and noisy measurements, $\underline{z}(t_i)$, representative of the "real world" stochastic processes. The tracking filter processes $\underline{z}(t_i)$ and outputs the state estimate $\hat{\underline{x}}(t_i)$. In line with the closed-loop operation of a pointing-tracking system, the filter also provides pointing commands to the truth model in the form of target position estimates. The simulated center of the FLIR field of view is then pointed to the predicted target location prior to the next sample time.

A Monte Carlo study keeps track of the filter estimation errors, $\underline{e}(t_i)$, over many runs of these sample-by-sample simulations. The statistics of these errors, means and standard deviations, then describe the filter's performance. As pointed out by Mercier, the trend of these error statistics as the number of simulation runs increases indicates how well the error sample statistics represents the true error statistical characteristics. In fact, he found that twenty simulation runs were sufficient to demonstrate convergence in the statistics. (Ref 13:p42) This was shown by plotting the standard deviation of the error at a number of chosen times versus the number of simulation runs. As the number of simulation runs increased, the standard deviation converged to a steady value. This same method was used to demonstrate convergence in this study's more dynamic environment. An



$\underline{x}(t_i)$	"real world" state values
$\underline{z}(t_i)$	"real world" measurements
$\hat{\underline{x}}(t_i)$	filter state estimate
$\underline{e}(t_i)$	filter estimation error
$\underline{w}(t_i)$	white Gaussian driving noise
$\underline{v}(t_i)$	white Gaussian measurement noise

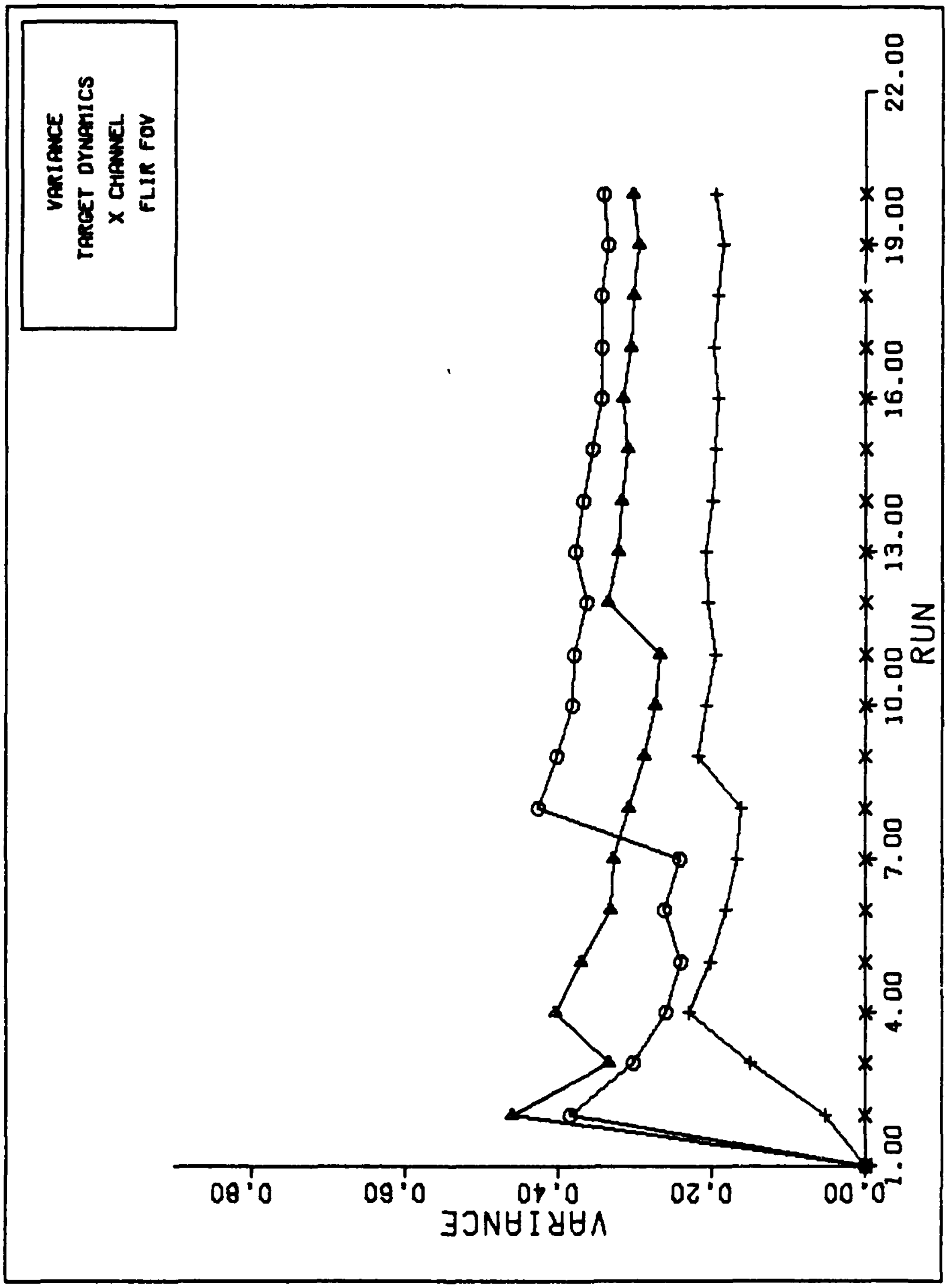
Monte Carlo Study
Figure 7

example of this convergence is shown in Fig. 8.

As mentioned in Section II, off-line tuning is required to obtain effective filter performance. Tuning consists of adjusting the correlation times and/or noise strengths in the filter model until Monte Carlo runs indicate effective performance. This study performed tuning by adjusting the discrete dynamic driving noise, $\underset{\sim}{Q}_{Fd}$, the target's maximum intensity, I_{max}, and the discrete measurement noise, $\underset{\sim}{R}_d$, within the filter model. Changing these values causes the filter's estimation error statistics to vary. In this study, effective performance is achieved by minimizing the error on all filter states simultaneously. Other problems may dictate other performance indicators such as minimizing the error on only specified states.

Agreement between the filter's estimate of the variance of the error with the actual error variance for each state is one indication of proper tuning. When these values closely match over the time period of the simulation, the actual error committed by the filter should be minimized. If the filter is underestimating or overestimating the true variance, the filter gains will not be optimal, and larger true errors result. Another indicator for tuning the EKF is trying to match the filter-computed variance with the Monte Carlo correlation (variance + mean^2), since the EKF has the potential for producing biased estimates.

Four target trajectories, each describing different air-to-air missile dynamics, are used in the performance



VARIANCE CONVERGENCE
Figure 8

analysis. Defined in an inertial reference frame whose origin was located at the tracker, the trajectory equations are summarized in Table 1. For a detailed development of the entire truth model see Appendix A.

The first trajectory, a constant velocity cross range path, was primarily used during software verification and robustness studies. Extended Kalman filter performance for this trajectory was compared to a similar test conducted by Harnly and Jensen to verify correct software implementation of the baseline EKF and the entire Monte Carlo simulation. Since the linear dynamics model (2-19) is well suited to this benign trajectory (Ref 6:p114), nonlinear measurement effects due to a turning missile's acceleration are eliminated. Thus, this trajectory is appropriate for evaluating filter robustness to other mismodelling. Those checked in this study were imperfect initial conditions, incorrect filter target intensity, I_{max}, and signal-to-noise ratios lower than assumed in the filter.

The remaining three trajectories are constant acceleration turns of 5, 10, and 20 g's. These are intended to be representative of air-to-air missile maneuvering. Since these turns are not well modelled by the filter, nonlinear effects appear which are not well addressed by the EKF. However, it is specifically with these maneuver effects that the alternative filters may demonstrate better performance.

Table 1

Truth Model Trajectories

Constant Velocity

$\dot{x}_e = -530.33$ m/sec $\qquad x_0 = 1000$ m

$\dot{y}_e = -530.33$ m/sec $\qquad y_0 = 1800$ m

$\dot{z}_e = 0$ m/sec $\qquad z_0 = 15,000$ m

Constant Acceleration Turns

Time ≤ 2 sec

$\dot{x}_e = -750$ m/sec $\qquad x_0 = 7000$ m

$\dot{y}_e = 0$ m/sec $\qquad y_0 = 500$ m

$\dot{z}_e = 0$ m/sec $\qquad z_0 = 15,000$ m

Time > 2 sec

$$\dot{x}_e = -750 \cos[\omega(t-\Delta T/2-2)]$$

$$\dot{y}_e = 750 \sin[\omega(t-\Delta T/2-2)]$$

$$\dot{z}_e = 0$$

$$x_e = 5500 - R\sin[\omega(t-2)]$$

$$y_e = 500 + R\{1-\cos[\omega(t-2)]\}$$

$$z_e = 15,000$$

	ω(rad/sec)	R(m)
5G	0.0653776	11,471.819
10G	0.1307553	5735.9037
20G	0.2615106	2867.9518

t = current time in simulation

ΔT = sample period

Computational Burden Analysis

Designing a filter algorithm for on-line implementation often involves tradeoffs between filter complexity and computer constraints. Although computer memory is not a critical constraint as it was in the past, the number of real time computations required is still a valid constraint, especially in systems with high data sample rates. With this in mind, the filters in this study, all of varying complexity, were analyzed to determine their computational burden.

In this study, the analysis was accomplished by simply counting the number of computations required to complete one filter cycle, i.e. one propagation period and one measurement update. This count included additions, multiplications, exponential evaluations, and number of inverses required. It must be noted that the counts in this study function only as an initial estimate of the computational burden. On-line implementable filter algorithms utilize additional means of reducing complexity such as exploiting symmetric or sparse matrices, transforming to canonical state space, or further model simplification. Also, to surmount on-line numerical difficulties, square root filtering techniques may be employed. Often these reintroduce some computational burden. (Ref 7:356,368)

V. Results

This section uses a comparitive approach in analyzing the alternative filters' performance and computational burden. Performance results from the extended Kalman filter (EKF) are the baseline for this study. The filters' best performance established under the assumed ideal conditions is presented first. To judge their performance in a more realistic envirnment, and to look at areas where EKF performance degrades, the ideal conditions are changed one at a time. Three areas are considered 1) filter response to imperfect initial conditions, 2) performance during low signal-to-noise ratio conditions, and 3) performance when the filter incorrectly models the maximum target intensity, I_{max}. Problems were encountered with this study's implementation of the statistically linearized filter which resulted in the filter being unable to track the target. These problems are addressed following the performance results. The final part of this section presents the results of the computational analysis.

Within this section, the performance criterion of maintaining track on the target is defined as the filter computed position remaining within three times the target size ($3\sigma_{g_1}$) of the true position. For any size target, this criterion means that the true image is close enough to the predicted image so that the residuals produce useful information. For cases of large σ_{g_1}, the $3\sigma_{g_1}$ boundary may well be off the 8-by-8 tracking window. Many of the results are tables of the average standard deviations and average absolute errors from Monte Carlo simulations. The values listed are time averages over the five second duration of the simulation run. As a final note of introduction, the individual case parameters and performance plots are contained in Appendix E for the EKF, Appendix F for the EKF with Bias Correction Term, and Appendix G for the Constant Gain EKF.

Best Performance

Looking first at the EKF with bias correction term, the results in Tables II and III show how closely it matches the performance of the baseline EKF. In fact, the additional second order bias correction term did not succeed in reducing the biases which occurred during the turn trajectories. Throughout the Monte Carlo study, as exemplified by the representative sample in Table IV, the bias correction term elements were consistently an order of magnitude smaller than the filter's predicted measurement. The second order information was thus deemed insignificant within the tracking filter, which resulted in no appreciable performance improvement.

As a result of this finding, only the constant gain EKF underwent a performance analysis, and not the associated constant gain EKF with bias correction term. The results for this filter are summarized in Table II. For the constant velocity trajectory case, performance matched the baseline EKF quite well, but the results were not as favor-

Table II

Performance Comparison
Ideal Conditions

	Extended Kalman Filter		EKF with Bias Correction Term		Constant Gain EKF	
Case: Constant Veloctity	E1		F1		G1	
Avg Standard Deviation	x	y	x	y	x	y
Position (pixels)						
Actual	0.17	0.17	0.17	0.17	0.17	0.17
Filter	0.20	0.20	0.20	0.20		
Velocity (pixels/sec)						
Actual	1.2	1.2	1.2	1.2	1.3	1.2
Filter	1.8	1.8	1.8	1.8		
Average Absolute Error						
Position (pixels)	0.04	0.04	0.04	0.04	0.04	0.04
Velocity (pixels/sec)	0.3	0.3	0.3	0.3	0.3	0.3
Case: 5G Turn	E2		F2		G2	
Avg Standard Deviation	x	y	x	y	x	y
Position (pixels)						
Actual	0.19	0.15	0.18	0.13	0.29	0.16
Filter	0.22	0.19	0.21	0.19		
Velocity (pixels/sec)						
Actual	1.4	1.2	1.3	1.1	2.8	1.3
Filter	2.1	1.9	1.9	1.7		
Average Absolute Error						
Position (pixels)	0.05	0.07	0.07	0.09	0.14	0.07
Velocity (pixels/sec)	1.2	2.3	1.3	2.6	1.6	2.4

Table III

Performance Comparison
Ideal Conditions

	Extended Kalman Filter		EKF with Bias Correction Term	
Case: 10G Turn	E3		F3	
Avg Standard Deviation	x	y	x	y
Position (pixels)				
Actual	0.19	0.16	0.18	0.16
Filter	0.21	0.19	0.21	0.19
Velocity (pixels/sec)				
Actual	1.6	1.3	1.5	1.3
Filter	2.1	1.9	2.1	1.9
Average Absolute Error				
Position (pixels)	0.08	0.13	0.08	0.13
Velocity (pixels/sec)	1.2	4.4	2.3	4.7
Case: 20G Turn	E6		F6	
Avg Standard Deviation	x	y	x	y
Position (pixels)				
Actual	0.20	0.16	0.20	0.16
Filter	0.23	0.20	0.23	0.20
Velocity (pixels/sec)				
Actual	2.6	2.4	2.6	2.4
Filter	4.5	4.1	4.5	4.1
Average Absolute Error				
Position (pixels)	0.04	0.05	0.04	0.05
Velocity (pixels/sec)	2.4	7.0	2.5	7.0

Table IV

Bias Correction Term and Filter Measurement Comparison

0.136 0.642	0.146 0.663	0.149 0.654	0.145 0.617	0.136 0.556	0.120 0.478	0.103 0.393	0.084 0.309
0.307 3.882	0.346 4.137	0.374 4.213	0.386 4.099	0.381 3.810	0.360 3.384	0.324 2.872	0.279 2.329
-0.057 11.982	-0.034 13.176	-0.001 13.844	0.040 13.899	0.083 13.332	0.122 12.219	0.151 10.700	0.167 8.952
-0.721 18.869	-0.825 21.411	-0.894 23.212	-0.915 24.045	-0.886 23.799	-0.810 22.506	-0.698 20.335	-0.567 17.556
-0.322 15.161	-0.428 17.750	-0.528 19.857	-0.608 21.224	-0.657 21.675	-0.667 21.150	-0.639 19.719	-0.577 17.566
0.286 6.215	0.305 7.508	0.309 8.667	0.297 9.558	0.272 10.072	0.235 10.141	0.194 9.756	0.151 8.967
0.212 1.300	0.252 1.620	0.286 1.930	0.310 2.196	0.321 2.388	0.318 2.481	0.302 2.462	0.273 2.335
0.046 0.139	0.057 0.178	0.068 0.219	0.078 0.257	0.084 0.289	0.088 0.309	0.087 0.317	0.083 0.311

Case F-3 10 G Turn
Run #1 $t_i = 2.30$

(Bias Element)
(Measurement)

able for the turn trajectories. The constant gain matrix had been designed to work for both the 5G and 10G turns (see Appendix C). Although the constant gain filter maintained track for the 5G turn, its performance statistics showed a significant performance degradation from the baseline EKF, especially in the x direction. This same filter was unable to maintain track on any of the 20 Monte Carlo 10G turn simulation runs. No further constant gains were tried, such as using a workable 10G turn gain during the 5G turn.

Imperfect Initial Conditions

Both the EKF and EKF with bias correction term underwent an acquisition sequence for this portion of the study. The first part of this sequence was setting the initial covariance, $\underline{P}(0)$, to values of 25 pixels^2 for the dynamics position states, 200 pixels^2/sec^2 for the velocity states, 400 pixels^2/sec^4 for the acceleration states, and 0.2 pixels^2 for the atmospheric jitter states. These values represent the uncertainty of the initial conditions given the filter. The other portion of the acquisition sequence involved a variable $\underline{Q}_{Fd}$ matrix for the first 0.5 second of each simulation run. Values within $\underline{Q}_{Fd}$ were linearly decreased to 25% of their initial value during this acquisition time period. This acquisition sequence causes the filter gain to stay appropriately high for the 0.5 second normally required for acquisition, and was also used by Harnly and Jensen (Ref 6:p104). However, their $\underline{P}(0)$ matrix (based on uncertainties inherent in a realistic application with hand-off from a radar) resulted in initial transients on some simulations of this study which resulted in the filter losing track. The previous values were

$$\underline{P}(0)=\begin{bmatrix} 25 & & & & & & & \\ & 25 & & & & & & \\ & & 2000 & & & & & \\ & & & 2000 & & & & \\ & & & & 100 & & & \\ & & & & & 100 & & \\ & & & & & & 0.2 & \\ & & & & & & & 0.2 \end{bmatrix}$$

Thus, the values cited for this study were obtained empirically to get good acquisition performance, necessitating an alteration of the third through sixth diagonal entry.

One way to analyze the recovery from imperfect initial conditions is to compare the Monte Carlo mean error time history for the first eight sample times as is done in Table V. The initial conditions for the results in this table were

$\hat{x}_d(0) = 0.5$ pixels
$\hat{y}_d(0) = 0.5$ pixels
$\dot{x}_e(0) = 525$ m/sec
$\dot{y}_e(0) = 535$ m/sec
$\dot{z}_e(0) = 8$ m/sec

which represent a 1.5% error in the true initial conditions. Notice that the EKF with bias correction term has a larger position mean error at the second and third sample times, although it does reach the average absolute mean error of 0.04 pixels at the same time as the EKF. This larger bias can be understood by comparing the bias correction term

Table V

Initial Conditon Mismatch Recovery Comparison

X Channel
Dynamics Position Mean Error

Time	Extended Kalman Filter	Constant Gain EKF	EKF with Bias Correction Term
0.033	-0.223	-0.543	-0.223
0.066	-0.293	-0.536	-0.378
0.100	-0.247	-0.555	-0.329
0.133	-0.137	-0.515	-0.180
0.166	-0.082	-0.428	-0.100
0.200	0.006	-0.274	-0.002
0.233	0.019	-0.155	0.018
0.266	0.033	-0.054	0.034

X Channel
Velocity Mean Error

0.033	-15.975	-12.202	-15.975
0.066	-4.335	-8.824	-6.488
0.100	-1.026	-6.080	-1.876
0.133	1.152	-3.352	1.371
0.166	1.601	-0.738	2.168
0.200	2.339	1.894	2.911
0.233	1.722	3.704	2.209
0.266	1.256	4.933	1.617

Constant Velocity Trajectory

elements to the predicted filter measurement during the acquisition phase. The comparison of these values in Table VI shows how the filter takes several sample periods to sort out the true magnitudes of the terms. In contrast, the constant gain EKF takes much longer in Table V to reach the 0.04 pixels mean error criterion. This is expected, since this filter cannot take advantage of the acquisition sequence's influence on $\underline{P}(t_i^-)$ which helped place more emphasis on the initial measurements within the baseline EKF.

As mentioned before, Tables V and VI correspond to initial state values that constitute a 1.5% error in the true initial conditions. Simulation runs were also made for values constituting a 2.5% error. The results of these runs (cases E-7 and E-8 in Appendix E and cases F-7 and F-8 in Appendix F) showed that neither the EKF nor EKF with bias correction term was able to maintain track when subjected to just the errors in initial velocity. For the case of just initial position error, the EKF maintained track in all 20 simulation runs, whereas the EKF with bias correction term lost track in 2 of the 20 runs. This substantiates the previous degraded performance of the EKF with bias correction term in Table V. It also demonstrates that both of these filters are more sensitive to initial velocity errors. Due to time constraints, the constant gain EKF was not tested with the larger initial condition errors.

Table VI

Filter Acquisition with Bias Correction Term

(Bias Element) (Measurement)

Four center pixels of 8-by-8 tracking window shown

-13.43 24.32	-13.43 24.32
-13.43 24.32	-13.43 24.32

$t_i = 0.033$

-13.43 1.94	12.63 1.69
13.03 1.80	13.87 2.05

$t_i = 0.066$

-0.42 15.20	-0.20 14.80
-0.24 14.89	-0.46 15.29

$t_i = 0.100$

-1.24 20.73	-1.19 20.49
-1.10 20.13	-1.15 20.33

$t_i = 0.133$

-1.00 22.33	-0.98 22.13
-0.93 21.85	-0.95 21.95

$t_i = 0.166$

Low Signal-to-noise Ratio Performance

This study's signal-to-noise ratio, I_{max}/σ_N, is an indicator of the intensity of the target image relative to the rms background and FLIR measurement noise. To test its effect on filter performance, the signal-to-noise ratio was reduced from the ideal 12.5 to a lower value of 2 by both decreasing I_{max} and increasing σ_N. (Realistic signal-to-noise ratios for this problem are between 1 and 25.) The results of changing this operating condition are summarized in Table VII for both the baseline EKF and EKF with bias correction term tracking the constant velocity trajectory. Both filters again demonstrate identical performance, but notice how their performance significantly degrades for the $I_{max}=4$ cases. This degradation is caused by the shape of the intensity profile changing dramatically for $I_{max}=4$, which makes it hard for the filter to seek out good measurement information in the noisy data. The constant gain EKF was not tested for this set of conditions.

Incorrect Filter Modelling Performance

This situation was meant to test filter robustness (the filter's tracking ability when some portion of its internal model inaccurately portrays the "real world"). The maximum target intensity was chosen as the mismodelled parameter and the performance plots in Figs 9 and 10 help to emphasize the filter comparison. These figures depict the time history of the x_d mean error ± one standard deviation

Table VII

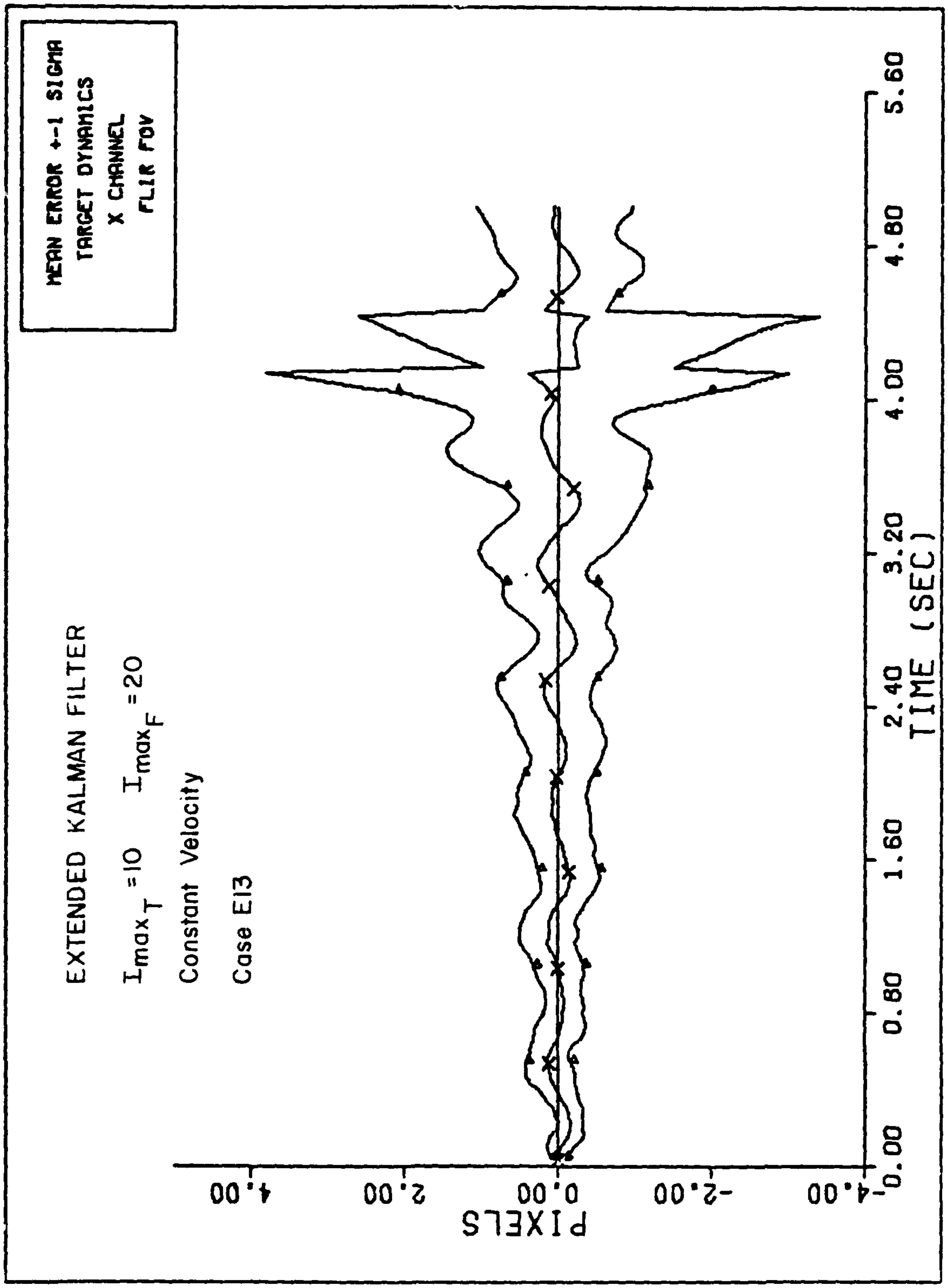
Performance Comparison
Low Signal-to-Noise Ratio, S/N = 2

	Extended Kalman Filter		EKF with Bias Correction Term	
Case: $I_{max}=4$, $\sigma_n=2$	E11		F11	
Avg Standard Deviation	x	y	x	y
Position (pixels)				
Actual	0.44	0.44	0.44	0.44
Filter	0.38	0.38	0.38	0.38
Velocity (pixels/sec)				
Actual	3.9	4.0	3.9	4.0
Filter	4.8	4.8	4.8	4.8
Average Absolute Error				
Position (pixels)	0.10	0.12	0.10	0.12
Velocity (pixels/sec)	0.8	1.0	0.8	1.0
Case: $I_{max}=25$, $\sigma_n=12.5$	E12		F12	
Avg Standard Deviation	x	y	x	y
Position (pixels)				
Actual	0.19	0.17	0.19	0.17
Filter	0.21	0.21	0.21	0.21
Velocity (pixels/sec)				
Actual	2.5	2.4	2.5	2.4
Filter	3.7	3.7	3.7	3.7
Average Absolute Error				
Position (pixels)	0.03	0.04	0.03	0.04
Velocity (pixels/sec)	0.5	0.5	0.5	0.5

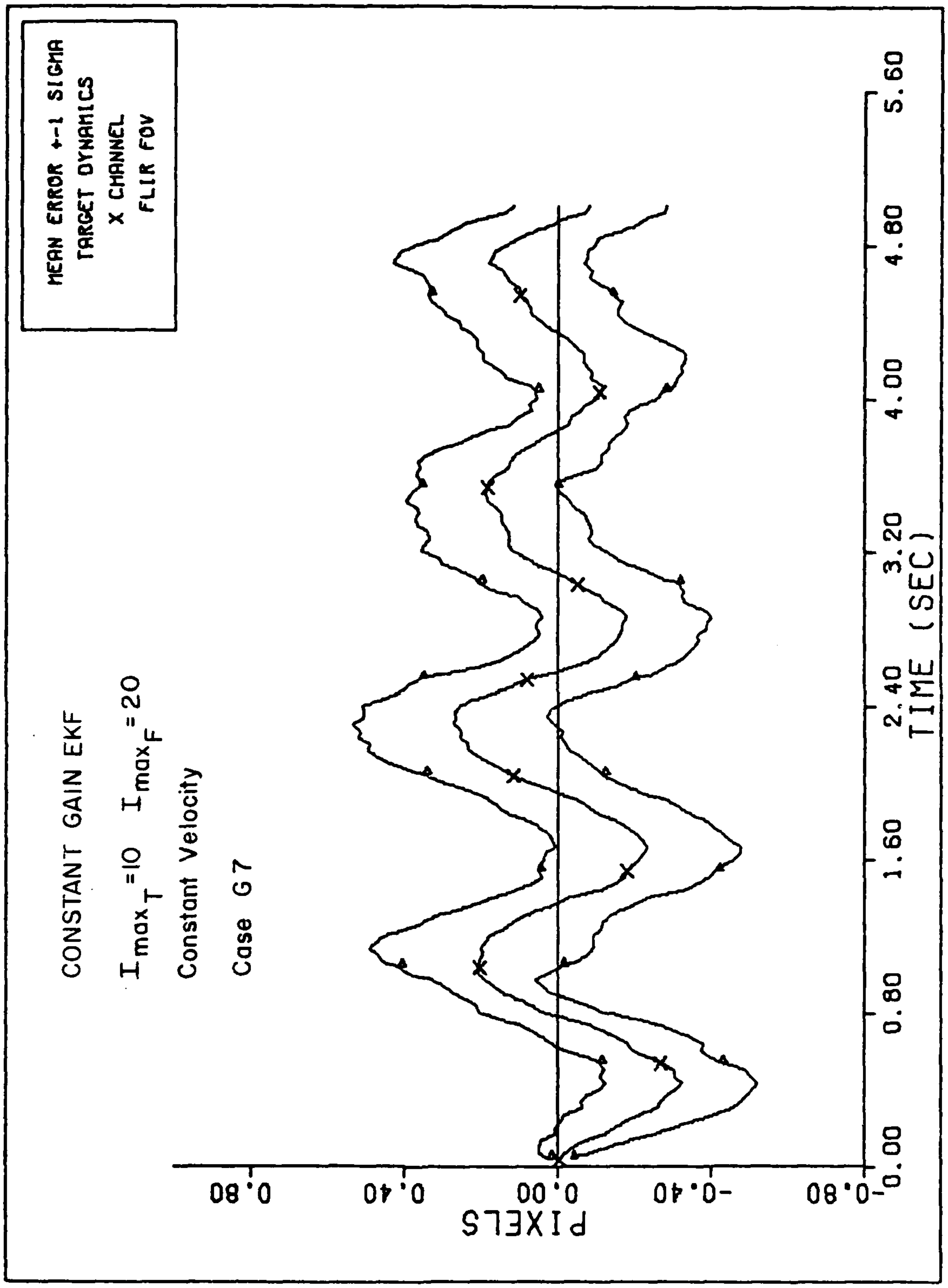
for the 20 Monte Carlo simulation runs. Fig 9 depicts the divergent performance of the EKF with I_{max} mismodelled as twice its truth model value. The two peaks after t=4 seconds represent loss of track in two of the runs (in computing statistics, error values are set to zero after loss of track occurs). Identical performance was recorded for the EKF with bias correction term. Fig 10 depicts the improved performance of the constant gain EKF. True to the claims in Section III, this filter is more robust to filter mismodelling of I_{max}. This results from the constant gain not being a function of the covariance calculations which contain the mismodelled filter approximation through the partial derivatives in $\underline{H}(t_i)$.

Statistically Linearized Filter

The statistically linearized filter developed in Section III was able to maintain track during the constant velocity trajectory with ideal conditions for only 1.1 seconds. This unacceptable performance is primarily attributed to the particular discretization used in this study. Particularly for the constant velocity case, the conditional probability density function, $f_{\underline{x}|\underline{z}}$, is tightly packed about the mean $\hat{\underline{x}}(t_i^-)$. Thus the values of P_ℓ in the eight pixels surrounding the pixel containing $[\hat{x}_{peak}(t_i), \hat{y}_{peak}(t_i)]$ approach zero. When this occurs, the matrix $\underline{H}(t_i)$ approaches $\underline{0}$ because of the filter discretization. Rewriting the expression (3-19)



X CHANNEL DYNAMICS ERROR (S/N= 5)
Figure 9



X CHANNEL DYNAMICS ERROR (S/N= 5)

Figure 10

$$\underline{\mathcal{H}}(t_i) = \{\widehat{\underline{h}[\underline{x}(t_i),t_i]\underline{x}^T(t_i)} - \widehat{\underline{h}[\underline{x}(t_i),t_i]}\,\hat{\underline{x}}^T(t_i^-)\}\;\underline{P}^{-1}(t_i^-)$$

note that when the value of P_ℓ for the center pixel equals one (due to small rms tracking error relative to pixel size), (3-19) changes to

$$\underline{\mathcal{H}}(t_i) = \{\underline{h}[\hat{\underline{x}}(t_i^-),t_i]\,\hat{\underline{x}}^T(t_i^-) - \underline{h}[\hat{\underline{x}}(t_i^-),t_i]\hat{\underline{x}}^T(t_i^-)\}\cdot$$

$$\underline{P}^-(t_i^-) = \underline{0}$$

A better approximation for determining $\underline{\mathcal{H}}(t_i)$ should result in the convergence of $\underline{\mathcal{H}}(t_i)$ to $\underline{H}(t_i)$ for this case.

To verify this condition, several simulation runs were performed using arbitrary values for the nine element P_ℓ discretization of $f_{\underset{\sim}{\underline{x}}|\underset{\sim}{\underline{z}}}$ within the filter. Table VIII summarizes the performance of these runs by comparing filter state estimates to those of the EKF and also with the truth model values. These values are taken at a single time point from a single simulation sample, and they are indicative of the problem's nature. For larger values of P_5, P_ℓ for the pixel containing $\hat{\underline{x}}(t_i^-)$, the performance degrades substantially, especially in estimating the acceleration. The associated $\underline{\mathcal{H}}(t_i)$ matrix elements also decreased in magnitude with larger values of P_5.

When the filter was allowed to compute its own P_ℓ values, simulation results showed that initially the P_ℓ values were essentially equal, as dictated by the large initial covariance $\underline{P}(0)$ described earlier. Subsequently, $f_{\underset{\sim}{\underline{x}}|\underset{\sim}{\underline{z}}}$ collapsed about $\hat{\underline{x}}(t_i^-)$ until the tenth sample period when P_5 reached a maximum of 0.94. At this point the

Table VIII

Statistically Linearized Performance with Varying $f_{\underset{\sim}{x}|\underset{\sim}{z}}$

State	Truth Model	Extended Kalman Filter	Statistically Linearized		
			P_5=0.984	P_5=0.928	P_5 = 0.840
x_d	-1940.082	-1940.055	-1937.496	-1939.787	-1940.033
y_a	-1913.741	-1914.069	-1906.856	-1914.316	-1914.042
$\dot{x}_d$	-1766.404	-1765.544	-1711.856	-1759.657	-1766.714
$\dot{y}_d$	-1751.593	-1754.177	-1650.729	-1725.654	-1742.793
$\ddot{x}_d$	—	0.8027	252.1689	-3.3211	3.9536
$\ddot{y}_d$	—	-27.6516	419.6761	108.6479	37.1561
x_a	0.08976	0.06197	-0.00016	0.00217	0.00321
y_a	-0.47767	-0.08391	-0.00015	-0.00425	-0.00507

Values at t_i=1.1 see in run #1 of Monte Carlo simulation.

discretization problem effects took hold and the filter's response was a spreading out of $f_{\underset{\sim}{\underline{x}}|\underset{\sim}{\underline{z}}}$. Within another ten sample periods P_s had reached a steady state value of 0.26.

Computational Burden

The results of this analysis are listed in Table IX, indicating the number of operations required for a single sample period. Compared to the baseline EKF, the EKF with bias correction term requires four times as many computations, whereas the constant gain EKF requires only one-tenth the computations. This study's implementation of the statistically linearized filter required twice as many computations as the baseline EKF, and this count is very much a function of the means chosen to approximate the evaluation of the conditional expectations inherent in the filter's structure. As mentioned in Section IV, these results should only be used as an estimate of what actual on-line computational requirements might be.

Table IX A
Computational Burden

Extended Kalman Filter	Adds	Multiplies	Exponentials	Inverses
State Propagation and Covariance Propagation	1408	1536		
H Partial Derivatives	320	1152		
h Measurement	512	896	64	
State Update	4160	5120		
Covariance Update	4096	4160		2
	10,496	12,864	64	2
Constant Gain EKF				
State Propagation	56	64		
h Measurement	512	896	64	
State Update	576	512		
	1144	1472	64	0

Table IX B
Computational Burden

Bxtended Kalman Filter with Bias Correction Term	Adds	Multiplies	Exponentials	Inverses
Extended Kalman Filter	10,496	12,864	64	2
Add Bias Correction Term	64			
Compute Bias Correction Term	29,504	35,520		
	40,064	48,384	64	2

Constant Gain EKF with Bias Correction Term	Adds	Multiplies	Exponentials	Inverses
Constant Gain EKF	1144	1472	64	
Add Bias Correction Term	64			
Compute Bias Correction Term	29,504	35,520		
	30,712	36,992	0	0

Table IX C

Computational Burden

Statistically Linearized	Adds	Multiplies	Exponentials	Inverses
State Propagation and Covariance Propagation	1408	1536		
$f_{\underset{\approx}{x} \mid \underset{\approx}{z}}$ Density Function	64	46	6	
$\underline{h}$ Measurement	800	1400	100	
$\hat{\underline{h}}$	576	576		
$\widehat{\underline{h}\underline{x}^T}$	4608	9216		
$\underline{H}$	4544	4608		
State Update	4160	5120		
Covariance Update	4096	4160		2
	20,204	26,662	106	2

VI. Conclusions and Recommendations

Conclusions

The scope of this study dictates making conclusions about each of the four alternative filters based on the results just presented. These conclusions are based on comparisons with the baseline extended Kalman Filter.

Extended Kalman Filter with Bias Correction Term. The EKF with bias correction term and its associated constant gain formulation do not present any noticeable performance improvement. Especially when considering the added computational burden of the bias correction term, these two filters are not worth implementing for on-line application.

Constant Gain Extended Kalman Filter. As predicted, the constant gain EKF displayed superior robustness, but overall performance was tied directly to the quality of the gain selection. In this study, performance while tracking maneuvering targets was not as good as the baseline EKF. The small computational loading is impressive and inviting for on-line implementation. Finally, a constant gain filter may require additional help in the acquisition phase due to its poorer incorrect initial condition recovery performance.

Statistically Linearized Filter. As implemented in this study, the statistically linearized filter was unsuccessful. This was due to the discretization used approximating the conditional probability density function, $f_{\underset{\sim}{\underline{x}}|\underset{\sim}{\underline{z}}}$. This filter was developed to gain improved performance when rms tracking errors for the EKF exceed one pixel. However, this level of error was never induced for long periods of time during the cases in this study. As a result, the particular discretization used would have heloped little to enhance performance even if it did not suffer from other problems.

Recommendations

Further research into these alternative filters should center about the constant gain extended Kalman filter and statistically linearized filter.

The small computational burden of the constant gain EKF is a desirable on-line attribute that could be exploited by looking for an "all purpose" gain. Perhaps defined by pole placement or entire eigenstructure assignment techniques, such a gain would be successful at tracking a maneuvering target not constrained to a specified trajectory. In light of this filter's poor acquisition performance, an on-line implementation might employ an extended Kalman filter computing its own gain for some specified acquisition phase. Another possible approach would be a scheduled gain structure. During acquisition or harsh maneuvers (detected by residual monitoring) a higher constant gain could be used, with the lower gain used for normal tracking.

The statistically linearized filter could be revamped by defining a new discretization scheme. This might include a variable scheme that switches from a one pixel defining area of $f_{\underset{\sim}{\underline{x}}|\underset{\sim}{\underline{z}}}$ to the nine pixel area used in this study depending upon the values of $\underline{P}(t_i^-)$. Regardless, the discretization should be smaller than one pixel to avoid the problem encountered in this study. Such a finer discretization however, must be weighed against the additional computational burden. It would be desirable to find an approximation such that $\underline{\mathcal{H}}(t_i)$ approximates $\underline{H}(t_i)$, instead of $\underline{0}$, when the probability that the true target intensity center lies within a single discrete area goes to one. Other alternatives for evaluations of the inherent conditional expectations should also be explored. There are several scenarios, such as targets performing jink maneuvers, that develop large enough $\underline{P}(t_i^-)$ values where this filter migh outperform the EKF.

Bibliography

1. "Advanced Adaptive Optics Control Techniques," Tech. Rept. TR-996-1, The Analytic Sciences Corp., Reading, Mass., Jan 1978.

2. "Advanced Adaptive Optics Control Techniques-Active Cavity Alignment," Tech. Rept. TR-78-169, The Analytic Sciences Corp., Reading, Mass., Jun 1978.

3. Athans, M., R.P. Wishner, and A. Bertolini, "Suboptimal State Estimators for Continuous-Time Nonlinear Systems from Discrete Noisy Measurement, "IEEE Transaction on Automatic Control, Vol AC-13, No. 5, pp 504-518, Oct 1968.

4. Hogge, C.B., and R.R. Butts, "Frequency Spectra for the Geometric Representation of Wavefront Distortions due to Atmospheric Turbulence, "IEEE Trans. Antenna and Propacat., Vol. AP-24, pp 144-154, Mar 1976.

5. Jazwinski, A.H., Stochastic Processes and Filtering Theory, Academic Press, New York, 1970.

6. Jensen, R.L., and D.A. Harnly, "An Adaptive Distributed-Measurement Extended Kalman Filter for a Short Range Tracker," M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, Dec 1979.

7. Maybeck, P.S., Stochastic Models, Estimation and Control, Vol. 1, Academic Press, New York, 1979.

8. Maybeck, P.S., Stochastic Models, Estimation and Control, Vol. 2 and 3, Academic Press, New York, 1982

9. Maybeck, P.S., "Design and Performance Analysis of an Adaptive Extended Kalman Filter for Target Image Tracking," to appear in "Advances in Theory and Technology of Application of Nonlinear Filters and Kalman Filters," NATO AGARDograph, NATO Advisory Group for Aerospace Research and Development, London, England, 1982.

10. Maybeck, P.S. and D.E. Mercier, "A Target Tracker Using Spatially Distributed Infrared Measurements," IEEE Trans. Automatic Control, Vol. AC-25, No. 2, pp 222-225, April, 1975.

11. Maybeck, P.S., D.A. Harnly and R.L. Jensen, "Robustness of a New Infrared Target Tracker," Proc. IEEE Nat. Aerospace and Electronics Conf., Dayton, Ohio, pp 639-644, May, 1980.

12. Maybeck, P.S., R.L. Jensen and D.A. Harnly, "An Adaptive Extended Kalman Filter for Target Image Tracking," *IEEE Trans. Aerospace and Electron. Sys.*, Vol. AES-17, No. 2, pp 173-180, March 1981.

13. Mercier, D.E., "An Extended Kalman Filter for Use in a Shared Aperture Medium Range Tracker," M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, Dec 1978.

14. Merritt, Paul H. "Beam Intensity Calculations for Jittered Beams," TR-78-174. Prepared for the Air Force Weapons Laboratory, Kirtland AFB, New Mexico, 1978.

15. Richards, C.L., "Correlation Tracking Algorithm," SAMRT-76-0076 Eng. Data Release, Aeronutronic Ford Corp., Newport Beach, Cal., July 2, 1976.

16. Richards, C.L., "Results of HAWK Image Tracking Experiment," SAMRT-76-0087 Eng. Data Release, Aeronutronic Ford Corp., Newport Beach, Cal., Aug 17, 1976.

17. Richards, C.L., "Correlation Tracking Software," SAMRT-76-0088 Eng. Data Release, Aeronutronic Ford Corp., Newport Beach, Cal., Aug 17, 1976.

18. Richards, C.L., "Precision Line and Point Reticle Location," SAMRT-77-0006 Tech. Data Release, Ford Aerospace and Commun. Corp., Newport Beach, Cal., Mar 8, 1977.

19. Safonov, M.G., and M. Athans, "Robustness and Computational Aspects of Nonlinear Stochastic Estimators and Regulators," *IEEE Trans. Automat. Control*, Vol. AC-23, No. 4, pp 717-725, Aug 1978.

Appendix A

Appendix A

Baseline Extended Kalman Filter and Truth Model

This appendix presents the additional equations, matrices, and other information used to construct the baseline EKF and truth model. The approach in this section is to present only enough detail to reconstruct the algorithms in this study. For the detailed development, the reader is asked to refer to the work of Mercier (Ref 13) and Harnly and Jensen (Ref 6).

Extended Kalman Filter

The state transition matrix associated with the propagation portion of the filter is

$$\underline{\Phi}_F = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & e^{\frac{-\Delta t}{\tau_A}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{\frac{-\Delta t}{\tau_A}} \end{bmatrix} \quad \text{(A-1)}$$

where Δt is the sample period and τ_A is the correlation time of the atmospheric jitter model. The discrete dynamic driving noise strength matrix is

$$\underline{Q}_{Fd} = \begin{bmatrix} \frac{\Delta t^5 \sigma_D{}^2}{20} & 0 & \frac{\Delta t^4 \sigma_D{}^2}{8} & 0 & \frac{\Delta t^3 \sigma_D{}^2}{6} & 0 & 0 & 0 \\ 0 & \frac{\Delta t^5 \sigma_D{}^2}{20} & 0 & \frac{\Delta t^4 \sigma_D{}^2}{8} & 0 & \frac{\Delta t^3 \sigma_D{}^2}{6} & 0 & 0 \\ \frac{\Delta t^4 \sigma_D{}^2}{8} & 0 & \frac{\Delta t^3 \sigma_D{}^2}{3} & 0 & \frac{\Delta t^2 \sigma_D{}^2}{2} & 0 & 0 & 0 \\ 0 & \frac{\Delta t^4 \sigma_D{}^2}{8} & 0 & \frac{\Delta t^3 \sigma_D{}^2}{3} & 0 & \frac{\Delta t^2 \sigma_D{}^2}{2} & 0 & 0 \\ \frac{\Delta t^3 \sigma_D{}^2}{6} & 0 & \frac{\Delta t^2 \sigma_D{}^2}{2} & 0 & \Delta t \sigma_D{}^2 & 0 & 0 & 0 \\ 0 & \frac{\Delta t^3 \sigma_D{}^2}{6} & 0 & \frac{\Delta t^2 \sigma_D{}^2}{2} & 0 & \Delta t \sigma_D{}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Q_7 \sigma_A{}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & Q_8 \sigma_A{}^2 \end{bmatrix} \qquad \text{(A-2)}$$

$$Q_7 = Q_8 = (1 - e^{-2\Delta t/\tau_a})$$

where $\sigma_D{}^2$ is the strength of the acceleration noise and $\sigma_A{}^2$ is the variance of the atmospheric jitter model output. As noted in Section II the covariance of the measurement model noise, $\underline{R}$, is computed as $(1/R_F)\underline{I}$. Adaptive size and shape estimation is accomplished through the equations

$$\hat{AR}_F(t_i) = \hat{AR}_F(t_{i-1}) + c \left. \frac{\partial L}{\partial AR} \right|_{AR = \hat{AR}_F} \qquad \text{(A-3)}$$

$$\hat{\sigma}_{g_1F}(t_i) = \hat{\sigma}_{g_1F}(t_{i-1}) + c \left. \frac{\partial L}{\partial \sigma_{g_1}} \right|_{\sigma_{g_1} = \hat{\sigma}_{g_1F}} \qquad \text{(A-4)}$$

where AR_F is defined as $\sigma_{g_1F}/\sigma_{g_2F}$, c is a constant determined empirically, and L is the quadratic cost function defined as

$$\begin{aligned} & L(\underline{x}(t_i), \quad \sigma_{g_1}, AR, \theta, t_i) = \\ & \quad [\underline{z}(t_i) - \underline{h}(\hat{\underline{x}}(t_i^-), t_i)]^T [\underline{z}(t_i) - \underline{h}(\hat{\underline{x}}(t_i^-), t_i)] \end{aligned} \qquad \text{(A-5)}$$

The angle θ is shown later in Fig A-2.

Truth Model

Trajectories. In developing the trajectories presented in Table I, a nominal velocity of 750 meters/second was chosen, which corresponds to a Mach number of 2.26 at standard temperature and pressure. These sample flight paths are all defined in the coordinate frame shown in Fig A-1.

The target's range from the tracker is

$$R = \sqrt{x_e^2 + y_e^2 + z_e^2} \tag{A-5}$$

and it's associated velocity is

$$V = \sqrt{\dot{x}_e^2 + \dot{y}_e^2 + \dot{z}_e^2} \tag{A-6}$$

Two more defining relationships which govern a constant accleration turn are

$$a = \omega^2 r \tag{A-7}$$

$$V = \omega r \tag{A-8}$$

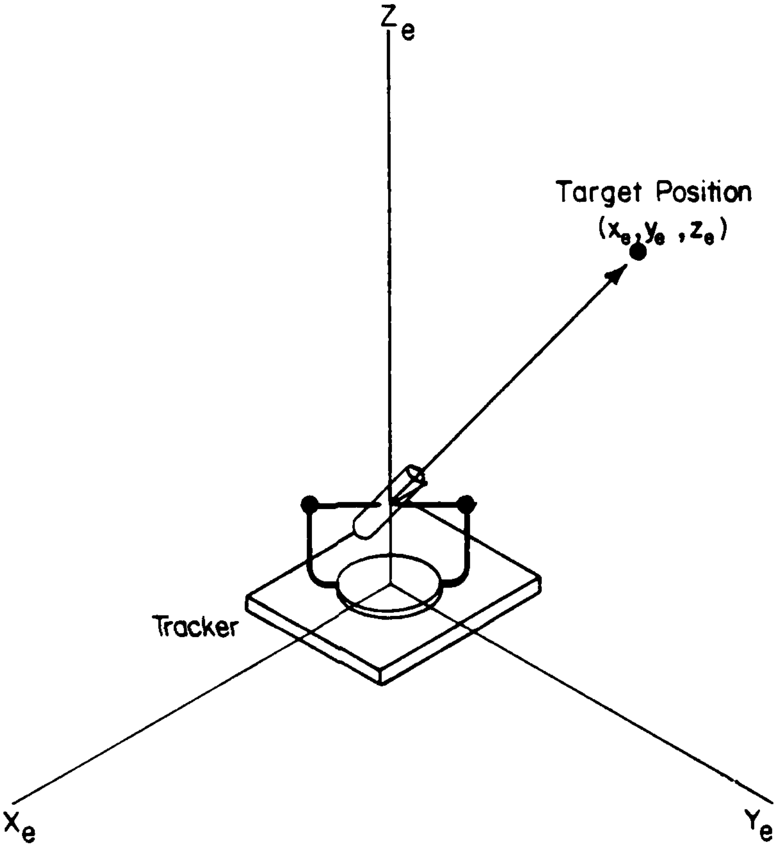
where a is the acceleration, ω the angular velocity, r the turn's radius, and V the same velocity as in (A-6). The trajectory equations were inserted into the Monte Carlo study through subroutine TRAJEC (see Appendix D).

For the constant velocity trajectory, the total velocity is split evenly between the x and y directions. From equation (A-6)

$$750 = \sqrt{\dot{x}_e^2 + \dot{y}_e^2}$$

$$|\dot{x}_e| = 530.33 \text{ m/sec}$$

$$|\dot{y}_e| = 530.33 \text{ m/sec}$$



Truth Model Reference Frame
Figure A-1

The trajectory path is at a constant altitude and brings the target close by the tracker to induce an inertial acceleration within the tracker filter's dynamics model. This path is described by

$$x_e(t) = x_e(0) - 530.33(t) \tag{A-9a}$$

$$y_e(t) = y_e(0) - 530.33(t) \tag{A-9b}$$

$$z_e(t) = z_e(0) \tag{A-9c}$$

With initial conditions of

$$x_e(0) = 1000 \text{ m}$$

$$y_e(0) = 1800 \text{ m}$$

$$z_e(0) = 15{,}000 \text{ m}$$

the maximum range is 15,140 meters and the minimum range is 15,010 meters during a five second run.

Again, the constant acceleration turn trajectory is constrained to the x-y plane at an altitude of 15,000 meters also. The target begins the run moving toward the tracker in the negative x direction. At time $t_i = 2$ seconds, the target begins the turn towards the positive y direction. For this constant acceleration turn, the velocity components during the turn are

$$\dot{x}_e(t) = V_2 \cos[\omega(t-2)] \tag{A-10a}$$

$$\dot{y}_e(t) = V_2 \sin[\omega(t-2)] \tag{A-10b}$$

$$\dot{z}_e(t) = 0 \tag{A-10c}$$

where V_2 is the velocity magnitude prior to starting the turn. The trajectory path during the turn is described by

$$x_e(t) = x_e(2) - r \sin [\omega(t-2)] \quad \text{(A-11a)}$$

$$y_e(t) = y_e(2) - r \{1 - \cos [\omega(t-2)]\} \quad \text{(A-11b)}$$

$$z_e(t) = z_e(2) \quad \text{(A-11c)}$$

where the parameters ω and r are evaluated using equation (A-7) and (A-8) at the desired acceleration level. An additional consideration is the sample data nature of this study. As a result, the truth model velocity at any sample time, t_i, should be representative of the average velocity over the previous sample period. For this study, the average velocity is assumed to be the velocity at the time half way between sample times. Equations (A-10a) and (A-10b) are then adjusted to

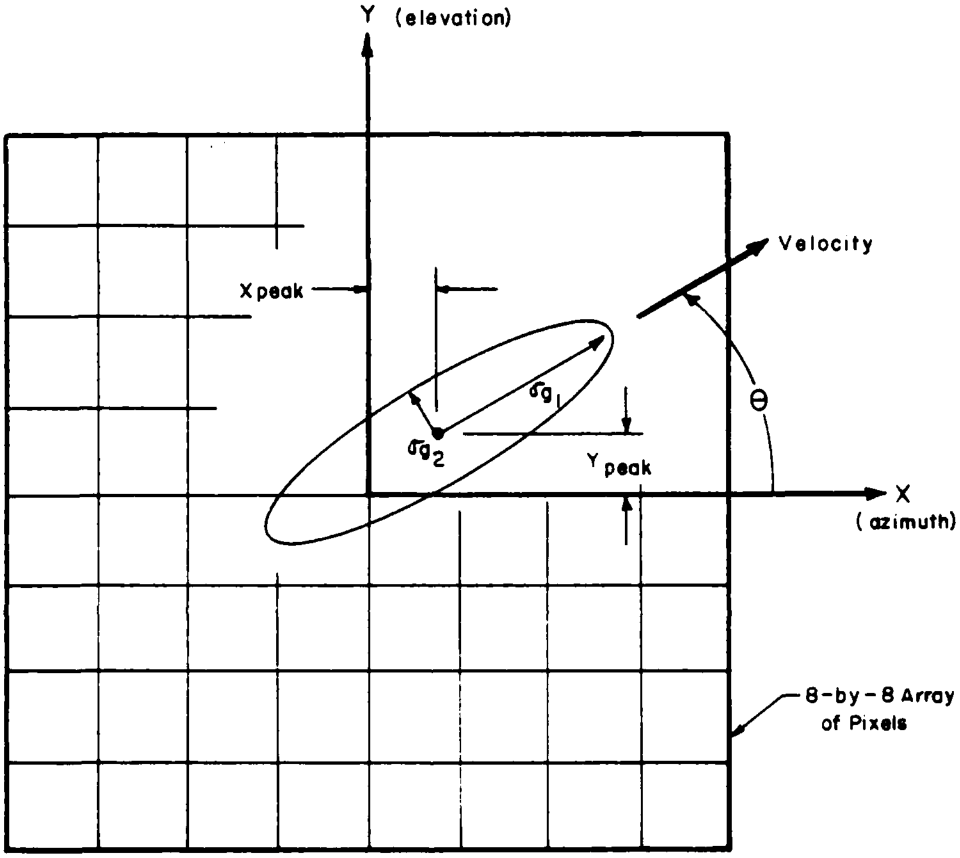
$$\dot{x}_e(t) = V_2 \cos [\omega(t-\Delta T/2-2)] \quad \text{(A-12a)}$$

$$\dot{y}_e(t) = V_2 \sin [\omega(t-\Delta T/2-2)] \quad \text{(A-12b)}$$

where ΔT is the sample period.

Transformation to FLIR image plane. Producing the missile image on the FLIR image plane from the position and velocity information defined in the inertial reference frame is accomplished by Subroutine XFORM (see Appendix D). The basic image orientation is shown in Fig A-2, where the x direction is defined as azimuth, α, and the y direction as elevation, β. The transformation from the tracker centered inertial frame to FLIR azimuth-elevation elements is accomplished by the equations

$$\dot{\alpha}(t) = \frac{z_e(t)\dot{x}_e(t) - x_e(t)\dot{z}_e(t)}{z_e(t)^2 + x_e(t)^2} \quad \text{(A-13)}$$



$$AR = \sigma_{g_1} / \sigma_{g_2}$$

Intensity Image Orientation

Figure A-2

$$\dot{\beta}(t) = \frac{r_h(t)\dot{y}_e(t) - y_e(t)\{[x_e(t)\dot{x}_e(t) + z_e(t)\dot{z}_e(t)]/r_h(t)\}}{\sqrt{x_e(t)^2 + y_e(t)^2 + z_e(t)^2}} \qquad \text{(A-14)}$$

where $r_h = [x_e(t)^2 + z_e(t)^2]^{1/2}$. The truth model uses a pixel as the unit of angular displacement rather that radians (1 pixels = 20 μrads). Therefore, $\dot{\alpha}(t)$ and $\dot{\beta}(t)$ are converted to pixels/second by dividing by 20 x 10^{-6} rads/pixel.

Truth Model Propagation. The complete truth model time propagation is defined by

$$\underline{x}(t_{i+1}) = \underline{\Phi}(t_{i+1},t_i)\underline{x}(t_i) + \left[\begin{array}{c} \underline{B}_d(t_i) \\ \hline \underline{0} \end{array}\right] \underline{u}_d(t_i) + \left[\begin{array}{c} \underline{0} \\ \hline \underline{c}\sqrt{Q_{Ad}} \end{array}\right] \underline{w}_A(t_i) \qquad \text{(A-15)}$$

where the state vector consists of two dynamic states and six atmospheric jitter states arrayed as

$$\underline{x} = [x_{DT}\ y_{DT}\ x_{AT_1}\ x_{AT_2}\ x_{AT_3}\ y_{AT_1}\ y_{AT_2}\ y_{AT_3}]^T \qquad \text{(A-16)}$$

the state transition matrix

$$\underline{\Phi} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{-a\Delta t} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{-b\Delta t} & \Delta t e^{-b\Delta t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-b\Delta t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{-a\Delta t} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-b\Delta t} & \Delta t e^{-b\Delta t} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-b\Delta t} \end{bmatrix} \quad \text{(A-17)}$$

with a and b the parameters of the third order atmospheric jitter model described in Section II by equation (2-5),

$$\underline{u}_d(t_i) = \begin{bmatrix} \dot{\alpha}(t_i + \Delta t/2) \\ \dot{\beta}(t_i + \Delta t/2) \end{bmatrix} \quad \text{(A-18)}$$

$$\underline{B}_d(t_i) = \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{(A-19)}$$

and $\underline{w}(t)$ is a vector of white Gaussian noises each with unit variance. The discrete atmospheric noise strength, $\underline{Q}_{Ad}$, has been decomposed with a Cholesky square root algorithm (Ref 7:p370) to form $\sqrt[c]{\underline{Q}_{Ad}}$. The entire discrete noise strength matrix, Q_{Td}, **developed in** Appendix D of Ref 6 using exact integration is implemented in subroutine TRUTHO (see Appendix D).

Truth Model Measurement. The truth model produces measurements using equaion (2-6). The integral is approximated by evaluating the function (2-16) at the midpoints of sixteen equal subdivisions of each pixel and averaging the results. These measurements are then corrupted by a 64 dimensional vector of noises, $\underline{v}(t_i)$, which has an associated noise variance matrix, $\underline{R}$, defined as

$$\underline{R} = \sigma_V{}^2 \begin{bmatrix} 1 & r_{1,2} & r_{1,3} & \cdots & r_{1,64} \\ r_{1,2} & 1 & r_{2,3} & \cdots & r_{2,64} \\ r_{1,3} & r_{2,3} & 1 & \cdots & r_{3,64} \\ \cdots & \cdots & \cdots & \ddots & \cdots \\ r_{1,64} & r_{2,64} & r_{3,64} & \cdots & 1 \end{bmatrix} \qquad \text{(A-20)}$$

where the coefficients $r_{m,n}$ indicated the correlation between noise componenets v_m and v_n at a given time t_i. Although this model allows for spatially correlated noise (Ref 6:p17-21), this study used a simple diagonal $\underline{R}$ matrix (all the noise components are uncorrelated) by setting the FORTRAN parameter ISPTL = 'NO'. The final truth model measurement then becomes

$$\underline{z}(t_i) = \underline{h}(\underline{x}(t_i), t_i) + \sqrt[c]{\underline{R}}\ \underline{w}(t_i) \qquad \text{(A-21)}$$

where $\underline{w}(t_i)$ is again a vector of independent white Gaussian noises each with unit variances.

Closed Loop. The truth model incorportates the filter's position estimates with the following equation

$$x_{peak_T}(t) = x_{DT} + x_{AT_1} + x_{AT_2} - x_{dF} \tag{A-22a}$$

$$y_{peak_T}(t) = y_{DT} + y_{AT_1} + y_{AT_2} - y_{dF} \tag{A-22b}$$

Appendix B

Appendix B

Bias Correction Terms - Partial Derivatives

Computing the bias correction terms in Section III requires evaluating the matrix of second partial derivatives $\partial^2 \underline{h}/\partial x_j \partial x_k$, with the function $\underline{h}[\hat{\underline{x}}(t_i),\ t_i]$ defined as

$$\underline{h}[\hat{\underline{x}}(t_i), t_i] = I_{max} \exp\{-\tfrac{1}{2}[(\frac{(x-x_p)\cos\theta + (y-y_p)\sin\theta}{\sigma_{g_1}})^2 + (\frac{(y-y_p)\cos\theta - (x-x_p)\sin\theta}{\sigma_{g_2}})^2]\} \qquad \text{(B-1)}$$

where

$x_p = x_{peak} = x_a + x_d$

$y_p = y_{peak} = y_a + y_d$

σ_{g_1} = standard deviation in direction of velocity vector

σ_{g_2} = standard deviation perpendicular to velocity vector

θ = angle which locates the velocity vector on the FLIR x-y coordinate frame (see Appendix A).

Notice that $\underline{h}[\hat{\underline{x}}(t_i^-), t_i]$ is only a function of four states, and that derivatives with respect to x_a and x_d are identical (results from $x_p = x_a + x_d$) as are those for y_a and y_d. With this in mind the partial derivatives follow.

$$\frac{\partial \underline{h}}{\partial x_p} = (PART_1)\ I_{max} \exp(ARG) \qquad \text{(B-2)}$$

where

$$ARG = \{-\tfrac{1}{2}[(\frac{(x-x_p)\cos\theta + (y-y_p)\sin\theta}{\sigma_{g_1}})^2 + (\frac{(y-y_p)\cos\theta - (x-x_p)\sin\theta}{\sigma_{g_2}})^2]\}$$

$$PART_1 = \{[\frac{(x-x_p)\cos\theta + (y-y_p)\sin\theta}{\sigma_{g_1}}][\frac{\cos\theta}{\sigma_{g_1}}] - [\frac{(y-y_p)\cos\theta - (x-x_p)\sin\theta}{\sigma_{g_2}}][\frac{\sin\theta}{\sigma_{g_2}}]\}$$

$$\frac{\partial \underline{h}}{\partial y_p} = (PART_2)\ I_{max}\exp(ARG) \tag{B-3}$$

where

$$PART_2 = \{[\frac{(x-x_p)\cos\theta + (y-y_p)\sin\theta}{\sigma_{g_1}}]\ [\frac{\sin\theta}{\sigma_{g_1}}] + [\frac{(y-y_p)\cos\theta - (x-x_p)\sin\theta}{\sigma_{g_2}}]\ [\frac{\cos\theta}{\sigma_{g_2}}]\} \tag{B-4}$$

$$\frac{\partial^2 \underline{h}}{\partial x_p \partial x_p} = \{[\frac{-\cos^2\theta}{\sigma_{g_1}} - \frac{\sin^2\theta}{\sigma_{g_2}}] + (PART_1)^2\} I_{max}\exp(ARG) \tag{B-5}$$

$$\frac{\partial^2 \underline{h}}{\partial y_p \partial y_p} = \{[\frac{-\sin^2\theta}{\sigma_{g_1}} - \frac{\cos^2\theta}{\sigma_{g_2}}] + (PART_2)^2\} I_{max}\exp(ARG)$$

$$\frac{\partial^2 \underline{h}}{\partial x_p \partial y_p} = \{[\frac{-\sin\theta\cos\theta}{\sigma_{g_1}} - \frac{\cos\theta\sin\theta}{\sigma_{g_2}}] + (PART_1)(PART_2)\} \cdot I_{max}\exp(ARG) \tag{B-6}$$

AD-A111 144   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 17/5
ALTERNATIVES TO AN EXTENDED KALMAN FILTER FOR TARGET IMAGE TRAC--ETC(U)
DEC 81 P R LEUTHAUSER
UNCLASSIFIED AFIT/GA/AA/81D-8 NL

2 of 3
AD
A111144

Appendix C

Appendix C

Selection of a Constant Gain

The technique chosen to establish $\underline{K}_{ss}$ for the turn trajectories was the linear combination of the gains obtained in Monte Carlo simulations defined by

$$\underline{K}_{ss} = 0.7\ \underline{K}_{5}(t_i=3.4)+0.3\underline{K}_{10}(t_i=3.4) \qquad (C\text{-}1)$$

where

$\underline{K}_{5}(t_i)$ = 5 G turn gain time history

$\underline{K}_{10}(t_i)$ = 10G turn gain time history

For the constant velocity trajectory, the gain chosen for $\underline{K}_{ss}$ was $\underline{K}(t_i=2.633)$, the position in the trajectory which is closest to the tracker. These two constant gain matrices are listed in Tables C-I and C-II.

Table C-I

Turn Trajectory $\underline{K}^{T}_{ss}$

.378E-3	-.519E-3	.436E-1	-.680E-1	.101E-1	-.168E-1	.117E-2	-.206E-2
.105E-2	-.194E-2	.120	-.254	.280E-1	-.624E-1	.328E-2	-.768E-2
.147E-2	-.429E-2	.174	-.559	.411E-1	-.137	.488E-2	-.168E-1
.845E-3	-.556E-2	.111	-.722	.272E-1	-.176	.335E-2	-.217E-1
-.139E-3	-.419E-2	-.800E-4	-.542	.150E-2	-.133	.372E-3	-.163E-1
-.390E-3	-.183E-2	-.354E-1	-.235	-.737E-2	-.577E-1	-.752E-3	-.707E-2
-.179E-3	-.461E-3	-.176E-1	-.593E-1	-.383E-2	-.144E-1	-.414E-3	-.177E-2
-.375E-4	-.671E-4	-.379E-2	-.851E-2	-.837E-3	-.209E-2	-.918E-4	-.255E-3
.317E-3	-.305E-3	.353E-1	-.405E-1	.814E-2	.100E-1	.940E-3	-.124E-2
.947E-3	-.127E-2	.107	-.168	.247E-1	-.415E-1	.287E-2	-.511E-2
.144E-2	-.313E-2	.166	-.409	.390E-1	-.100	.458E-2	-.124E-1
.856E-3	-.466E-2	.108	-.580	.262E-1	-.142	.319E-2	-.174E-1
-.244E-3	-.368E-2	-.130E-1	-.476	-.157E-2	-.116	-.370E-5	-.142E-1
-.544E-3	-.176E-2	-.532E-1	-.225	-.113E-1	-.552E-1	-.121E-2	-.675E-2
-.268E-3	-.484E-3	-.271E-1	-.617E-1	-.596E-2	-.150E-1	-.654E-3	-.183E-2
-.602E-4	-.766E-4	-.619E-2	-.967E-2	-.137E-2	-.236E-2	-.152E-3	-.287E-3
.245E-3	-.139E-3	.269E-1	-.188E-1	.618E-2	-.470E-2	.708E-3	-.585E-3
.309E-3	-.656E-3	.891E-1	-.875E-1	.205E-1	-.217E-1	.236E-2	-.269E-2
.132E-2	-.179E-2	.149	-.236	.346E-1	-.552E-1	.399E-2	-.720E-2
.807E-3	-.265E-2	.969E-1	-.370	.231E-1	-.910E-1	.275E-2	-.111E-1
-.375E-3	-.260E-2	-.306E-1	-.334	-.601E-2	-.817E-1	-.558E-3	-.998E-2
-.748E-3	-.136E-2	-.758E-1	-.174	-.165E-1	-.425E-1	-.181E-2	-.518E-2
-.382E-3	-.415E-3	-.397E-1	-.523E-1	-.886E-2	-.127E-1	-.982E-3	-.153
-.930E-4	-.726E-4	-.967E-2	-.904E-2	-.217E-2	-.218E-2	-.243E-3	-.264E-3
.181E-3	-.264E-4	..195E-1	-.409E-2	.445E-2	-.106E-2	.506E-3	-.137E-3
.654E-3	-.153E-3	.708E-1	-.221E-1	.161E-1	-.564E-2	.183E-2	-.717E-3
.115E-2	-.502E-3	.126	-.690E-1	.288E-1	-.175E-1	.330E-2	-.216E-2
.707E-3	-.937E-3	.793E-1	-.123	.183E-1	-.304E-1	.212E-2	-.373E-2
-.516E-3	-.966E-3	-.517E-1	-.122	-.113E-1	-.309E-1	-.124E-2	-.376E-2
-.959E-3	-.608E-3	-.100	-.749E-1	-.226E-1	-.180E-1	-.253E-2	-.217E-2
-.525E-3	-.212E-3	-.556E-1	-.254E-1	-.125E-1	-.607E-2	-.141E-2	-.723E-3
-.138E-3	-.425E-4	-.146E-1	-.499E-2	-.330E-2	-.117E-2	-.373E-3	-.138E-3
.127E-3	.377E-4	.135E-1	.441E-2	.305E-2	.103E-2	.345E-3	.122E-3
.501E-3	.193E-3	.531E-1	.231E-1	.119E-1	.548E-2	.135E-2	.653E-3
.955E-3	.567E-3	.100	.697E-1	.225E-1	.168E-1	.252E-2	.201E-2
.572E-3	.953E-3	.580E-1	.120	.127E-1	.294E-1	.140E-2	.359E-2
-.647E-3	.916E-3	-.728E-1	.120	-.168E-1	.297E-1	-.196E-2	.367E-2
-.116E-2	.501E-3	-.126	.691E-1	-.289E-1	.161E-1	-.331E-2	.216E-2
-.693E-3	.156E-3	-.739E-1	.226E-1	-.168E-1	.578E-2	-.192E-2	.734E-3
-.195E-3	.274E-4	-.210E-1	.427E-2	-.480E-2	.111E-2	-.546E-3	.143E-3
.855E-4	.651E-4	.895E-2	.811E-2	.200E-2	.195E-2	.224E-3	.236E-3
.365E-3	.382E-3	.378E-1	.492E-1	.845E-2	.116E-1	.938E-3	.142E-2
.744E-3	.129E-2	.752E-1	.165	.165E-1	.401E-1	.182E-2	.489E-2
.426E-3	.252E-2	.366E-1	.324	.735E-2	.795E-1	.714E-3	.956E-2
.746E-3	.283E-2	-.902E-1	.368	-.218E-1	.907E-1	-.258E-2	.111E-1
-.133E-2	.183E-2	-.150	.242	-.347E-1	.597E-1	-.403E-2	.737E-2
-.842E-3	.683E-3	-.929E-1	.919E-1	-.213E-1	.228E-1	-.246E-2	.282E-2
-.246E-3	.149E-3	-.289E-2	.203E-1	-.664E-2	.506E-2	-.760E-3	.629E-3
.552E-4	.683E-4	.567E-2	.871E-2	.126E-2	.212E-2	.140E-3	.258E-3
.254E-3	.446E-3	.257E-1	.568E-1	.567E-2	.138E-1	.623E-3	.169E-2
.549E-3	.166E-2	.531E-1	.213	.114E-1	.522E-1	.122E-2	.638E-2
.289E-3	.359E-2	.182E-1	.463	.277E-2	.113	.146E-3	.139E-1
-.796E-3	.446E-2	-.102	.579	-.247E-1	.142	-.302E-2	.174E-1
-.144E-2	.321E-2	-.166	.419	-.390E-1	.107	-.458E-2	.127E-1
-.982E-3	.134E-2	-.111	.176	-.256E-1	.436E-1	-.278E-2	[illegible]
-.399E-3	.336E-3	-.377E-1	.436E-1	-.869E-2	.108E-2	-.101E-2	.133E-2
.342E-4	.600E-4	.346E-2	.766E-2	.764E-2	.186E-2	.837E-4	.227E-3
.169E-3	.424E-3	.167E-1	.544E-2	.363E-2	.133E-1	.373E-3	.163E-2
.368E-3	.173E-2	.354E-1	.222	.741E-2	.545E-1	.761E-3	.667E-2
.175E-3	.406E-2	.446E-2	.526	-.506E-3	.128	.248E-3	.158E-1
.789E-3	.554E-2	-.104	.719	-.262E-1	.176	-.319E-2	.216E-1
.146E-2	.438E-2	-.173	.571	-.411E-1	.140	-.438E-2	.172E-1
-.107E-2	.203E-2	-.124	.266	-.290E-1	.655E-1	-.339E-2	.803E-1
-.412E-3	.557E-3	.465E-1	.732E-1	-.107E-1	.190E-1	-.125E-2	.222E-2

Table C-II

Constant Velocity Turn Trajectory $\underline{K}^T_{ss}$

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	.001	-.003	0.0	-.001	0.0	0.0
0.0	0.0	0.0	-.02	0.0	-.005	0.0	-.0007
0.0	-.001	-.02	-.07	-.005	-.02	-.0008	-.002
-.001	-.002	-.09	-.2	-.02	-.05	-.003	-.006
-.003	-.004	-.2	-.3	-.06	-.08	-.009	-.01
-.004	-.004	-.3	-.3	-.09	-.09	-.01	-.01
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	.002	-.002	.0005	-.0005	0.0	0.0
0.0	0.0	.007	-.01	.002	-.004	0.0	-.0005
0.0	-.0007	.003	-.05	.0005	-.01	0.0	-.002
0.0	-.002	-.05	-.1	-.01	-.04	-.002	-.005
-.007	-.003	-.2	-.2	-.05	-.07	-.007	-.009
-.004	-.004	-.3	-.3	-.08	-.08	-.01	-.01
-.004	-.003	-.3	-.2	-.08	-.07	-.01	-.009
0.0	0.0	.003	-.001	.0009	0.0	0.0	0.0
0.0	0.0	.01	-.007	.003	-.002	.0005	0.0
0.0	0.0	.03	-.03	.008	-.008	.001	-.001
0.0	-.001	.02	-.09	.004	-.02	0.0	-.003
-.0008	-.002	-.07	-.2	-.02	-.04	-.003	-.006
-.003	-.003	-.2	-.2	-.06	-.06	-.008	-.008
-.003	-.003	-.3	-.2	-.07	-.05	-.009	-.006
-.002	-.001	-.2	-.1	-.05	-.03	-.007	-.004
0.0	0.0	.02	0.0	.005	0.0	.001	0.0
.0007	0.0	.05	-.005	.01	-.0008	.002	0.0
.001	0.0	.08	-.02	.02	-.005	.003	-.0005
.0007	-.0007	.05	-.05	.01	-.01	.002	-.002
-.001	-.001	-.08	-.08	-.02	-.02	-.003	-.003
-.002	-.0009	-.2	-.08	-.04	-.02	-.006	-.003
-.002	-.0005	-.1	-.05	-.04	-.01	-.005	-.002
-.001	0.0	-.07	-.02	-.02	-.006	-.003	.0009
.001	0.0	.07	.02	.02	.005	.002	.0008
.002	0.0	.1	.04	.04	.01	.005	.002
.002	.0008	.2	.07	.04	.02	.006	.003
.0009	.0009	.07	.07	.02	.02	.002	.002
-.0007	.0007	-.05	.04	-.01	.01	-.002	.002
-.001	0.0	-.09	.02	-.02	.004	-.003	0.0
-.0008	0.0	-.06	.003	-.02	0.0	-.002	0.0
0.0	0.0	-.02	0.0	-.005	0.0	-.007	0.0
.002	.001	.2	.09	.05	.03	.007	.003
.003	.002	.3	.2	.07	.05	.009	.006
.003	.003	.2	.2	.05	.06	.008	.008
.0008	.002	.07	.2	.02	.05	.003	.006
0.0	.002	-.02	.09	-.004	.02	-.0005	.003
0.0	0.0	-.03	.03	-.008	.008	-.001	.001
0.0	0.0	-.01	.007	-.004	.002	-.0005	0.0
0.0	0.0	-.004	.001	-.0009	0.0	0.0	0.0
.004	.003	.3	.2	.08	.06	.01	.008
.004	.004	.3	.3	.08	.08	.01	.01
.002	.003	.2	.3	.05	.07	.007	.009
.0005	.002	.05	.2	.01	.04	.002	.005
0.0	.0007	-.004	.06	-.0006	.02	0.0	.002
0.0	0.0	-.007	.02	-.002	.004	0.0	.0005
0.0	0.0	-.002	.003	-.0006	.0006	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.004	.004	.3	.3	.09	.09	.01	.01
.003	.004	.2	.3	.06	.08	.009	.01
.001	.002	.1	.2	.03	.05	.004	.006
0.0	.001	.02	.07	.006	.02	.0009	.003
0.0	0.0	0.0	.02	0.0	.005	0.0	.0007
0.0	0.0	-.001	.003	0.0	.0009	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Appendix D

Appendix D

Computer Software

This appendix contains the computer code for the algorithms presented in Section II and III along with the additional program modules required to perform the Monte Carlo study. Programing was done using FORTRAN 77 implemented on a CDC 6600 computer system. Program LEUTER performed the executive function for controlling the Monte Carlo simulation runs. First developed by Mercier, program PLOT was the executive for generating the Monte Carlo error statistics and associated CALCOMP plots. (Ref 13) Each program and/or subroutine contains a brief description of its function.

```
      PROGRAM LEUTER
*
C          LEUTER IS A MONTE CARLO SIMULATION PROGRAM FOR PERFORMANCE
C     ANALYSIS OF TARGET TRACKING FILTERS WHICH USE FLIR DATA.  MUCH
C     OF THE TRUTH MODEL DEVELOPMENT IS TAKEN FROM PREVIOUS THESES AT
C     AFIT.  HIGHER ORDER FILTERS WILL BE ANALYZED AND COMPARED TO AN
C     EXTENDED KALMAN FILTER IN LASER POINTING AND TRACKING.
*

*
      REAL Q(3,3),IMAX,WORK(3,3),SAVE(14),QFD1(8,8),QD(8,8),RN(64,64)
*
      INTEGER NPS,NFS,NPIX,ONE
*
      CHARACTER ISPTL*3
*
      COMMON/IN/ NRUN,TFINAL,SIGS1,SIGAT,C(5),SIGF10,SIGF2,AR0,FIMAX0,
     !           SN,SIGMF0,RF,SIGMAB
      COMMON/FLIR/ XFOV,YFOV,NPIX,CSTH,SNTH,SIGMS,SIGFLR,ASPRO,
     !             IMEAS,VMAX,RANGE0,RANGE,IMAX,SIGVF,SIGPVF,AR
      COMMON/ARRAY/ XFP(8),XFPO(8),XFM(8),PFP(8,8),PFPOLD(8,8),
     !             FPFP(8,8),PFM(8,8),QFD(8,8),EXTRA(8,8),PHIF(8,8),
     !             PHIFT(8,8),UT(2,1),PL2P(64,2),ED(8,2),TEMP(8,8),
     !             TEMP1(8,8),SQQD(8,8),W(8),XS(8),H(64,8),HF(9,9),
     !             Z(8,8),R(64,64),WKAREA(50,50),HT(8,64),PHI(9,9)
      COMMON/CHANGE/RIH(64,1),DX(8),GAIN(8,64)
*
C     ...VARIABLE DEFINITIONS...
*
C     ...EXECUTE LEUTER...
*
*

*
C     ...OPEN FILE FOR SAVING PLOT DATA...
*
      OPEN(8,FILE='MYDATA',STATUS='NEW',FORM='UNFORMATTED',IOSTAT=IOS)
      IF (IOS .LT. 0) THEN
         PRINT*,'FILE MYDATA WILL NOT OPEN'
         GO TO 99
      ENDIF
*
C     ...READ INPUT DATA...
*
      READ*,NRUN,TFINAL
      READ*,SIGS1,SIGMAB,SIGFLR,SIGAT,SIGMS
      READ*,IMAX,ASPRO,ISPTL
      IF (ISPTL .EQ. 'YES') THEN
         READ*,C(1),C(2),C(3),C(4),C(5)
      ENDIF
      READ*,SIGF10,SIGMF0,SIGF2
      READ*,AR0,RF,FIMAX0
*
C     ...VERIFY AND ECHO DATA...
*
      FIMAX=ABS(FIMAX0)
      SIGVF=ABS(SIGMF0)
      CALL INCHEK(ISPTL)
*
```

```
*
         CALL CHOLEX(RN,R,64)
         PRINT'(///2X,A/)','THE CHOLESKY SQUARE ROOT OF RN IS:'
         CALL MWRITE(R)
      ELSE
         DO 22 I=1,64
         DO 22 J=1,64
            R(I,J)=0.0
22       CONTINUE
         DO 11 I=1,64
            R(I,I)=1.0
11       CONTINUE
      ENDIF
*
C     ...FILTER...
*
      CALL FILTRM(DT,PHIF,PHIFT)
      PRINT'(///2X,A/)','THE FILTER STATE TRANSITION MATRIX IS:'
      CALL MOUT(PHIF)
*
*
      PRINT'(A)','1'
C
      PRINT *,'********** BEGIN THE MONTE CARLO SIMULATION **********'
C
*
*
C     ...TOP OF RUN LOOP...
*
      DO 999 L=1,NRUN
      TIME=0.0
      JPRINT=0
*
      PRINT'(/////)'
      PRINT*,'**********  RUN NUMBER ',L,' ******** *'
*
C     ...RUN INITIALIZATION...
*
*
*
C     ...ACQUISITION PHASE FILTER QD MATRIX...
*
      CALL QAQUIR(DT,SIGF10,SIGF2,QFD)
*
C     ...INITIAL COVARIANCE P+(0)...
*
      CALL COVARI(PFP)
*
C     ...PROVIDE FILTER WITH INITIAL VELOCITY AND POSITION
*
      XFP(1)=0.0
      XFP(2)=0.0
      CALL XFORM(TIME,XFP(3),XFP(4),XVEH,YVEH,ZVEH,VMAX,RANGE)
      XFP(5)=0.0
      XFP(6)=0.0
      XFP(7)=0.0
      XFP(8)=0.0
```

```
C      ...DEFINE PARAMETERS...
*
C          SAMPLE RATE IS 1/30 TH OF A SECOND
       DT = 1./30.

*
C      ...INITIALIZE TRUTH MODEL VARIABLES...
*
       CALL RANSET(75682)
       ONE = 1
       NPS = 8
       NPIX=8
       NMS = NPIX*NPIX
       NFS=8
       NFS2=NFS*NFS

       NIS = 3
       RI=1./RF
       XFOV=8.
       YFOV=8.
*
*
C      ...PROBLEM INITIALIZATION...
*
*
*
C      ...TRUTH MODEL...
*
       CALL TRUTHM(DT,SIGAT,SIGS1,PHI,BD,QD,Q)
       PRINT'(///2X,A/)','THE TRUTH MODEL STATE TRANSITION MATRIX IS:'
       CALL MOUT(PHI)
       PRINT'(///2X,A/)','THE TRUTH MODEL DISCRETE INPUT MATRIX IS:'
       CALL MOUT1(BD)
       PRINT'(///2X,A/)','THE TRUTH MODEL QD MATRIX IS:'
       CALL MOUT(QD)
C
C      ...TAKING CHOLESKY SQUARE ROOT OF QD...
C
       SQQD(1,1) = SQRT(QD(1,1))
       SQQD(2,2) = SQQD(1,1)
       CALL CHOLEX(Q,WORK,NIS)
       DO 33 I=1,NIS
       DO 33 J=1,NIS
       SQQD(I+2,J+2) = WORK(J,I)
       SQQD(I+5,J+5) = WORK(J,I)
    33 CONTINUE
       PRINT'(///2X,A/)','THE CHOLESKY SQUARE ROOT OF QD IS:'
       CALL MOUT(SQQD)
*
C      ...FLIR SPATIAL BACKGROUND NOISE...
*
       IF (ISPTL .EQ. 'YES') THEN
          CALL RNOIZE(C,SIGMAB,RN,R)
          PRINT'(///2X,A/)','64 X 64 SPATIAL CORRELATION MATRIX:'
          CALL MWRITE(RN)
*
C         ...TAKING CHOLESKY SQUARE ROOT OF RN...
```

```
*
C       ...PRINT INITIAL RUN CONDITIONS...
*
        IF (L .EQ. 1) THEN
           PRINT'(///T2,A/)','THE FILTER ACQUISITION QD MATRIX IS:'
           CALL MOUT1(QFD)
           PRINT'(///T2,A/)','THE INITIAL P(+) MATRIX IS:'
           CALL MOUT(PFP)
           PRINT'(///T2,A/)','THE INITIAL FILTER ESTIMATE XF(+) IS:'
           CALL MOUT2(XFP)
           PRINT'(///4(TR8,A5,TR8))',' XVEH',' YVEH',' ZVEH','RANGE'
           PRINT'(4(TR5,G16.10))',XVEH,YVEH,ZVEH,RANGE
        ENDIF
*
C       ...RESET OTHER INITIAL RUN CONDITIONS...
*
        DO 44 I=1,NPS
           XS(I)=0.0
44      CONTINUE
        XCENTR=0.
        YCENTR=0.
        FIMIN =0.
        IMEAS=0
        FIMAX=ABS(FIMAX0)
        AP=AP0
        RANGE0=RANGE
        SIGVF=ABS(SIGMF0)
        SIGF1=ABS(SIGF10)
*
*
C       ...TIME LOOP STARTS HERE...
*
*
    50  TIME = TIME + DT
        JPRINT=JPRINT+1
        IF (TIME .GT. TFINAL) GO TO 999
*
C       ...TRUTH MODEL STATE PROPAGATION...
*
        CALL XFORM(TIME,UT(1,1),UT(2,1),XVEH,YVEH,ZVEH,VMAX,RANGE)
        CALL TRUTH(XCENTR,YCENTR,NPS,ONE)
*
C       ...FILTER PROPAGATION...
*
        CALL MOVEUP(NFS,NFS2,ONE)
*
C       ...FORM CENTROID POSITION AND FILL TRUTH ARRAY...
*
        XPEAK = XS(1) + XS(3) + XS(4)  -XFM(1)
        YPEAK = XS(2) + XS(6) + XS(7)  -XFM(2)
*
C       ...CHECK FOR FILTER LOSING TRACK...
*
        IF(ABS(XPEAK) .GT. 3.0*ASPRO*SIGMS) THEN
           PRINT*,'LOST TRACK, X CHANNEL, MEAS CALLED ',IMEAS,' TIMES.',
     +           '  RUN NUMBER ',L
           GO TO 60
```

```
      ELSEIF (ABS(YPEAK).GT. 3.*MASPRO*SIGMS) THEN
         PRINT*,'LOST TRACK, Y CHANNEL, MEAS CALLED ',IMEAS,' TIMES.',
     :            '  RUN NUMBER ',L
         GO TO 60
      ELSE
      ENDIF
*
C     ...TRUTH MODEL MEASUREMENT...
*
      CALL MEAS(XPEAK,YPEAK,4)
*
C     ...PERFORM MEASUREMENT UPDATE FOR THE FILTER...
*
      CALL MEASUF(RI,NFS,NMS,ONE,NFS2,FIMAX,SIGMF0,XPEAK,YPEAK)
*
C     ...PRINT RESULTS EVERY TENTH SECOND...
*
      IF((JPRINT .EQ. 3) .AND. (L .EQ. 1)) THEN
         PRINT'(/////)'
         PRINT*,'   TIME=   ',TIME,'   *********************************'
         PRINT'(///4(TR8,A5,TR8))',' XVEH',' YVEH',' ZVEH','RANGE'
         PRINT'(4(TR5,G16.10))',XVEH,YVEH,ZVEH,RANGE
         PRINT'(///2(TR6,A7,TR6))','UT(1,1)','UT(2,1)'
         PRINT'(2(TR5,G16.10))',UT(1,1),UT(2,1)
         PRINT'(///T2,A/)','THE TRUTH MODEL STATE IS:'
         CALL MOUT2(XS)
         PRINT'(///T2,A/)','THE UPDATED FILTER ESTIMATE XF(+) IS:'
         CALL MOUT2(XFP)
         PRINT'(///T2,A/)','THE UPDATED P(+) MATRIX IS:'
         CALL MOUT(PFP)
         PRINT'(///T2,A/)','THE TRANSPOSED GAIN MATRIX IS:'
         CALL MOUT3(GAIN)
         JPRINT=0
      ENDIF
*
      IF (SIGMF0.GT.0.0) GO TO 52
*
C     ...COMPUTE NEW ESTIMATE OF SIGVF AND AR...
*
      DO 51 I=1,NMS
      AR= AR+.001*(RIH(I,1)*PL2P(I,1))
      SIGVF=SIGVF+.001*(RIH(I,1)*PL2P(I,2))
51    CONTINUE
      IF(AR.LE.1.) AR=1.
      IF(SIGVF.LE.0.) SIGVF=SIGMF0
52    CONTINUE
*
C     ...RETUNE QFD MATRIX...
*
      CALL QTUNE(QFD,TIME,DT,SIGF1)
C
C     ...WRITE DATA TO FILE TAPE6...
C
      SAVE(1) = XS(1)
      SAVE(2) = UT(1,1)
      SAVE(3) = XS(2)
      SAVE(4) = UT(2,1)
      SAVE(5) = XFP(1)
      SAVE(6) = XFP(3)
      SAVE(7) = XFP(2)
      SAVE(8) = XFP(4)
      SAVE(9) = XCENTR
      SAVE(10)  =  YCENTR
      SAVE(11)  =  PFP(1,1)
      SAVE(12)  =  PFP(3,3)
      SAVE(13)  =  PFP(2,2)
      SAVE(14)  =  PFP(4,4)
      WRITE(8) (SAVE(I),I=1,14)
*
C     ...BOTTOM OF TIME LOOP...
      GO TO 50
*
*
C     ...FILTER LOST TRACK...
*
60    DO 110 J=1,14
 110  SAVE(J)=0.
      WRITE(8) (SAVE(I),I=1,14)
      TIME = TIME+DT
      IF (TIME .GT. TFINAL) GO TO 999
      GO TO 60
*
C     ...BOTTOM OF RUN LOOP...
999   CONTINUE
*
*
99    CONTINUE
      ENDFILE 8
      CLOSE(8)
      END
      SUBROUTINE MEASUR(RI,NFS,NMS,ONE,NFS2,FIMAX,SIGMFP,XPEAK,YPEAK)
*
C          MEASUR PERFORMS THE FILTER MEASUREMENT UPDATE BASED ON
C     EXTENDED KALMAN FILTER EQUATIONS FOR BOTH THE STATE VECTOR
C     COVARIANCE MATRIX.
*
C     ...VARIABLES...
*
      REAL RI,Z1(64),HF1(64),BIAS(64)
      INTEGER NFS,NMS,ONE,NFS2
*
      COMMON/CHANGE/RIH(64,1),DX(8),GAIN(8,64)
      COMMON/ARRAY/ XFP(8),XFPO(8),XFM(8),PFP(8,8),PFPOLD(8,8),
     &             PPFP(8,8),PFM(8,8),DFD(8,8),EXTRA(8,8),PHIF(8,8),
     &             PHIFT(8,8),UT(2,1),PL2P(64,2),BD(8,2),TEMP(8,8),
     &             TEMP1(8,8),SQOD(8,3),W(8),XS(8),H(64,8),HF(8,8),
     &             Z(8,8),R(64,64),WKAREA(50,51),HT(8,64),PHI(8,8)
*
C     ...BEGIN UPDATE...
*
      CALL SHIFTA(PFPOLD,PFP,NFS2,NFS2)
      XPEAK = XFM(7)
      YPEAK = XFM(8)
      IF((XFM(3)**2+XFM(4)**2).EQ.0.) XFM(3)=.001
*
C     ...FORM FILTER CENTROID POSITION AND FILL OUT NON LINEAR
C        SMALL H. CALCULATE PARTIAL SMALL H PARTIAL X...
*
      CALL MEASF(XPEAK,YPEAK,ONE,FIMAX,SIGMFP,BIAS)
      DO 60 I=1,NMS
      DO 60 J=1,NFS
      HT(J,I) = H(I,J)
   60 CONTINUE
*
C     ...THE INVERSE COVARIANCE FORM IS USED BECAUSE OF THE
C        LARGE MEASUREMENT VECTOR SIZE...
*
      CALL MMPY(PFP,HT,H,NFS,NMS,NFS)
C        PFP= HT * H  (R-1 IS SCALAR AND MULTIPLIED LATER)
      IDGT = 0
      CALL LINV2F(PFM,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
C        EXTRA= P(TI-)-1
      DO 65 I=1,NFS
      DO 65 J=1,NFS
      PFP(I,J)=PFP(I,J)*RI + EXTRA(I,J)
   65 CONTINUE
C        PFP= P(TI+)-1= HT * R-1 * H + P(TI-)-1
      IDGT = 0
      CALL LINV2F(PFP,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
      DO 70 I=1,NFS
      DO 70 J=1,NFS
      PFP(I,J) = EXTRA(I,J)
   70 CONTINUE
*
C     ...CHECK FOR NEGATIVE DIAGONALS...
*
      DO 130 I=1,NFS
      IF(PFP(I,I).GT.0.) GO TO 130
      PRINT*,'PFP(I,I)= ',PFP(I,I)
      PFP(I,I)=PFPOLD(I,I)
 130  CONTINUE
C        PFP=P(TI+)
C
*
C     ...REARRANGE MEASUREMENTS TO VECTORS...
*
      DO 80 J=1,NFS
      DO 80 K=1,NFS
         HF1(K+((J-1)*8)) = HF(K,J)
         Z1(K+((J-1)*8)) = Z(K,J)
80    CONTINUE

*
C     ...PERFORM STATE VECTOR UPDATE USING BIAS CORRECTION TERMS...
*
      CALL MADD(RIH,Z1,HF1,NMS,ONE,NFS)
C        RIH=RESIDUALS= Z-SMALL H
      CALL MADD(HF1,RIH,BIAS,NMS,ONE,NFS)
C        HF1 = RESIDUALS - BIAS CORRECTION TERMS
      CALL MMPY(GAIN,PFP,HT,NFS,NFS,NMS)
C           GAIN = P(TI+) * HT
      DO 85 I=1,NFS
      DO 85 J=1,NFS2
         GAIN(I,J)=GAIN(I,J)*R1
85    CONTINUE
C           GAIN = P(TI+) * HT * R-1
      CALL MMPY(DX,GAIN,HF1,NFS,NMS,ONE)
C        DX = P(TI+) * HT * R-1 * (Z - SMALL H - BIAS)
      CALL MADD(XFP,DX,XFM,NFS,ONE,ONE)
C        XFP=X(TI+)= P(TI+) * R-1 * HT * (Z - SMALL H) + X(TI-)
      END
      SUBROUTINE MEAS (XPEAK,YPEAK,ISUB)
*
C     ...VARIABLES...
*
      REAL IMAX,W1(64,1),WD(64,1)
      COMMON/FLIR/ XFOV,YFOV,NPIX,CSTH,SNTH,SIGMS,SIGFLR,ASPRO,
     !             IMEAS,VMAX,RANGE0,RANGE,IMAX,SIGVF,SIGPVF,AR
      COMMON/ARRAY/ XFP(8),XFPO(8),XFM(8),PFP(8,8),PFPOLD(8,8),
     !             PPFP(8,8),PFM(8,8),QFD(8,8),EXTRA(8,8),PHIF(8,8),
     :             PHIFT(8,8),UT(2,1),PL2P(64,2),BD(8,2),TEMP(8,8),
     :             TEMP1(8,8),SQQD(8,8),W(8),XS(8),H(64,8),HF(8,8),
     !             Z(8,8),R(64,64),WKAREA(50,50),HT(8,64),PHI(8,8)
*
C     ...TRUTH MEASUREMENT...
*
      ZMIN = 0.
      SIGPV=SIGMS*RANGE0/RANGE
      PLVEL=SQRT(UT(1,1)**2+UT(2,1)**2)
      SNTH=UT(2,1)/PLVEL
      CSTH=UT(1,1)/PLVEL
      SIGV=(1.+(ASPRO-1.)*PLVEL/VMAX)*SIGPV
      I = (NPIX*ISUB)
      IDIV = ISUB**2
      XINCR = XFOV/FLOAT(I)
      YINCR = YFOV/FLOAT(I)
      X = -1.*XFOV/2. + XINCR/2.
      Y = YFOV/2. - YINCR/2.
      X0 = X
      CALL NOISE(64,W1)
      CALL MMPY(WD,R,W1,64,64,1)
      MN=0
      DO 20 K=1,NPIX
      DO 15 J=1,NPIX
      MN=MN+1
      TOTAL = 0.
      XN = X
      YN = Y
      DO 10 N=1,ISUB
      YCSTH=(YN-YPEAK)*CSTH
      YSNTH=(YN-YPEAK)*SNTH
      DO 5 M =1,ISUB
      ARGSPV=YCSTH-(XN-XPEAK)*SNTH
      ARGSV=(XN-XPEAK)*CSTH+YSNTH
      ARG=-((ARGSV/SIGV)**2+(ARGSPV/SIGPV)**2)*.5
      TOTAL =TOTAL+EXP(ARG)*IMAX
      YN = X +FLOAT(M)*XINCR
    5 CONTINUE
      YN = Y- FLOAT(N)*YINCR
      XN = X
   10 CONTINUE
      Z(K,J)=TOTAL/FLOAT(IDIV)
C
C        ADD BACKGROUND AND FLIR NOISE BOTH ZERO MEAN
C
      IF(SIGFLR.EQ.0.) GO TO 30
      GAUSS=0.
      DO 110 LL=1,12
      GAUSS=GAUSS+RANF()
  110 CONTINUE
      F=(GAUSS-6.)*SIGFLR
      Z(K,J) = Z(K,J) + F
 30   Z(K,J)=Z(K,J)+WD(MN,1)
      IF(Z(K,J).LT.ZMIN)ZMIN=Z(K,J)
      X = XO +FLOAT(ISUB)*XINCR
   15 CONTINUE
      Y = Y - FLOAT(ISUB)*YINCR
      X=XO
   20 CONTINUE
      IMEAS=IMEAS+1
      IF(ZMIN.EQ.0.)RETURN
      DO 25 I=1,NPIX
      DO 25 J=1,NPIX
      Z(I,J) = Z(I,J)-ZMIN+.1
   25 CONTINUE
      RETURN
      END




      SUBROUTINE MEASF(XPEAK,YPEAK,ISUB,FIMAX,SIGMFR,BIAS)
*
C     ...VARIABLES...
*
      REAL IMAX,XPEAK,YPEAK,FIMAX,SIGMFR,BIAS(64)
      COMMON/FLIR/ XFOV,YFOV,NPIX,CSTH,SNTH,SIGMS,SIGFLR,ASPRO,
     !             IMEAS,VMAX,RANGE0,RANGE,IMAX,SIGVF,SIGPVF,AR
      COMMON/ARRAY/ XFP(8),XFPO(8),XFM(8),PFP(8,8),PFPOLD(8,8),
     !             PFFP(8,8),PFM(8,8),QFD(8,8),EXTRA(8,8),PHIF(8,8),
     !             PHIFT(8,8),UT(2,1),P_2P(64,2),BD(8,2),TEMP(8,8),
     !             TEMP1(8,8),SQQD(8,8),W(8),XS(8),H(64,8),HF(8,8),
     !             Z(8,8),R(64,64),WKAREA(50,50),HT(8,64),PHI(8,8)
*
C     ...FILTER MEASUREMENT...
*
      ZMIN = 0.
      PLVEL= SQRT(XFM(3)**2+XFM(4)**2)
      SNTH= XFM(4)/PLVEL
      CSTH= XFM(3)/PLVEL
      SIGPVF=SIGVF/AR
      I = (NPIX*ISUB)
      IDIV = ISUB**2
      XINCR = XFOV/FLOAT(I)
      YINCR = YFOV/FLOAT(I)
      X = -1.*XFOV/2. + XINCR/2.
      Y = YFOV/2. - YINCR/2.
      XO = X
      DO 30 II=1,NPIX
      DO 30 JJ=1,NPIX
         TEMP(II,JJ)=0.0
 30   CONTINUE
```

```
      DO 2  K=1,NPIX
      NUM = K
      DO 15 J=1,NPIX
      TOTAL = 0.
      SUM1=0.
      SUM2=0.
      SUM3=0.0
      SUM4=0.0
      SUM5=0.0
      XN = X
      YN = Y
      DO 10 N=1,ISUB
      YCSTH=(YN-YPEAK)*CSTH
      YSNTH=(YN-YPEAK)*SNTH
      DO 5 M =1,ISUB
      ARGSPV=YCSTH-(XN-XPEAK)*SNTH
      ARGSV=(XN-XPEAK)*CSTH+YSNTH
      ARG=-((ARGSV/SIGVF)**2+(ARGSPV/SIGPVF)**2)*.5
      PART= EXP(ARG)*FIMAX
      TOTAL = TOTAL+PART
      PART1=(ARGSV*CSTH/SIGVF**2-ARGSPV*SNTH/SIGPVF**2)
      PART2=(ARGSPV*CSTH/SIGPVF**2+ARGSV*SNTH/SIGVF**2)
      SUM1 = SUM1+PART*PART1
      SUM2 = SUM2+PART*PART2
      SUM3 = SUM3-PART*(CSTH**2/SIGVF**2+SNTH**2/SIGPVF**2)+
     $       (PART1*PART1*PART)
      SUM4 = SUM4-PART*(SNTH**2/SIGVF**2+CSTH**2/SIGPVF**2)+
     $        (PART2*PART2*PART)
      SUM5 = SUM5+PART*(-CSTH*SNTH/SIGVF**2+CSTH*SNTH/SIGPVF**2)+
     $       (PART2*PART1*PART)
      XN = X +FLOAT(M)*XINCR
    5 CONTINUE
      YN = Y-FLOAT(N) * YINCR
      XN = X
   10 CONTINUE
      HF(K,J) = TOTAL/FLOAT(IDIV)
      IF(SIGMFF.GT.0.) GO TO 16
      PL2P(K+(J-1)*8,1)= PART*(-(ARGSPV/SIGVF)**2*AR)
      PL2P(K+(J-1)*8,2)= PART*((ARGSV**2+ARGSPV**2*AR**2)/SIGVF**3)
15    CONTINUE
      IF(HF(K,J).LT.ZMIN) ZMIN=HF(K,J)
      H(NUM,1) = SUM1/FLOAT(IDIV)
      H(NUM,2) = SUM2/FLOAT(IDIV)
      H(NUM,3)=0.
      H(NUM,4)=0.
      H(NUM,5)=0.
      H(NUM,6)=0.
      H(NUM,7)=H(NUM,1)
      H(NUM,8)=H(NUM,2)
      TEMP(1,1) = SUM3/FLOAT(IDIV)
      TEMP(1,2) = SUM5/FLOAT(IDIV)
      TEMP(2,1) = TEMP(1,2)
      TEMP(2,2) = SUM4/FLOAT(IDIV)
      TEMP(7,7) = TEMP(1,1)
      TEMP(7,8) = TEMP(1,2)
      TEMP(8,7) = TEMP(2,1)
      TEMP(8,8) = TEMP(2,2)
      CALL BIASCT(BIAS(NUM),TEMP,TEMP1,PFM,NPIX)
      X = XN +FLOAT(ISUB)*XINCR
      NUM = NUM + NPIX
   15 CONTINUE
      Y = Y - FLOAT(ISUB)*YINCR
      X=XN
   20 CONTINUE
      IF(ZMIN.EQ.0.)RETURN
      DO 25 I=1,NPIX
      DO 25 J=1,NPIX
      HF(I,J)=HF(I,J)-ZMIN+0.1
   25 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BIASCT(BITERM,TEMP,TEMP1,PFM,NPIX)
C
C          BIASC COMPUTES THE INDIVIDUAL BIAS CORRECTION TERM
C     USING THE SECOND PARTIAL DERIVATIVE MATRIX CONTAINED IN TEMP.
C
C
C     ...VARIABLES...
C
      REAL BITERM,TEMP(8,8),TEMP1(8,8),PFM(8,8)
C
C     ...CALCULATE BIAS CORRECTION TERMS...
C
      BITERM=0.
      CALL MMPY(TEMP1,TEMP,PFM,NPIX,NPIX,NPIX)
      DO 1  I=1,NPIX
         BITERM = BITERM+TEMP1(I,I)
1     CONTINUE
      BITERM=0.5*BITERM
C
C        BITERM=1/2*TRACE(DH2/DX2 * P(-) )
C
      END
      SUBROUTINE STATLN(XPEAK,YPEAK,NFS,NMS,ONE)
*
C         STATLN COMPUTES THE REQUIRED CONDITIONAL EXPECTATIONS AND
C     MATRICES REQUIRED FOR FILTER MEASUREMENT UPDATE.
*
C     ...VARIABLES...
*
      REAL CPDFX(3),CPDFY(3),CPDF(3,3),NORM,XPEAK,YPEAK,TEMP2(64,8),
     !    HXHAT(64,8)
      INTEGER NFS,NMS,ONE
*
      COMMON/ARRAY/ XFP(8),XFPO(8),XFM(8),PFP(8,8),PFPOLD(8,8),
     !             PFFP(8,8),PFM(8,8),QFD(8,8),EXTRA(8,8),PHIF(8,8),
     !             PHIFT(8,8),UT(2,1),PL2P(64,2),BD(8,2),TEMP(8,8),
     !             TEMP1(8,8),SOOD(8,3),W(8),XS(8),H(64,8),HF(8,8),
     !             Z(8,8),R(64,64),WKAREA(53,53),HT(8,64),PHI(8,8)
      COMMON/CHANGE/FIH(64,1),DX(8),GAIN(8,64),HSL(10,10)
*
C     ...INITIALIZE VARIABLES...
*
      DO 5 I=1,NMS
      DO 5 J=1,NFS
         HXHAT(I,J)=0.0
5     CONTINUE
      DO 10 I=1,NFS
      DO 10 J=1,NFS
         HF(I,J)=0.0
10    CONTINUE
      NORM=0.0
*
C     ...COMPUTE VALUES FOR CONDITIONAL PROBABILITY DENSITY FUNCTION...
*
*
C     ...DETERMINE XPEAK SIGMA**2 AND YPEAK SIGMA**2...
*
      XPSIG=PFM(1,1)+PFM(7,1)+PFM(1,7)+PFM(7,7)
      YPSIG=PFM(2,2)+PFM(8,2)+PFM(2,8)+PFM(8,8)
*
C     ...FIND PIXEL CONTAINING XPEAK AND YPEAK...
*
      IF ((ANINT(XPEAK)-XPEAK) .LT. 0.0) THEN
         X0=ANINT(XPEAK)+0.5
      ELSE
         X0=ANINT(XPEAK)-0.5
      ENDIF
*
      IF ((ANINT(YPEAK)-YPEAK) .LT. 0.0) THEN
         Y0=ANINT(YPEAK)+0.5
      ELSE
         Y0=ANINT(YPEAK)-0.5
      ENDIF
*
C     ...COMPUTE VALUES FOR X DIRECTION...
*
      DO 10 I=-1,1
         ARGX=-0.5*((((X0+FLOAT(I))-XPEAK)**2)/XPSIG)
         CPDFX(I+2)=(1.0/SQRT(XPSIG))*EXP(ARGX)
```

```
            ARGY=-0.5*((((Y0+FLOAT(I))-YPEAK)**2)/YPSIG)
            CPDFY(I+2)=(1.0/SQRT(YPSIG))*EXP(ARGY)
15       CONTINUE
*
C        ...EVALUATE AND NORMALIZE PIXEL VALUES OF CPDF...
*
         DO 25 I=1,3
         DO 25 J=1,3
            CPDF(I,J)=CPDFX(I)*CPDFY(J)
            NORM=NORM+CPDF(I,J)
25       CONTINUE
         DO 30 I=1,3
         DO 30 J=1,3
            CPDF(I,J)=CPDF(I,J)/NORM
30       CONTINUE
         PRINT*
         PRINT*,'THE STATLN CPDF IS:'
         CALL MOUT(CPDF)
*
C        ...COMPUTE CONDITIONAL EXPECTATION OF MEASUREMENT,HHAT...
C                  HF = HHAT
*
         DO 45 I=1,NFS
         DO 45 J=1,NFS
            DO 40 K=1,3
            DO 40 L=1,3
               HF(I,J)=HF(I,J)+HSL(I+L-1,J-K+3)*CPDF(K,L)
40          CONTINUE
45       CONTINUE
*
C        ...COMPUTE CONDITIONAL EXPECTATION OF H * X, HXHAT...
*
*
C        ...FIND PERCENTAGE OF ONE PIXEL SHIFT IN XHAT IS DUE TO
C                     DYNAMICS AND THAT DUE TO ATMOSPHERIC EFFECTS...
*
         DENOM1=SQRT(PFM(1,1))+SQRT(PFM(7,7))
         DENOM2=SQRT(PFM(2,2))+SQRT(PFM(8,8))
         DELXDY=SQRT(PFM(1,1))/DENOM1
         DELXAT=SQRT(PFM(7,7))/DENOM1
         DELYDY=SQRT(PFM(2,2))/DENOM2
         DELYAT=SQRT(PFM(8,8))/DENOM2
*
C        ...FIND HXHAT...
*
         DO 50 I=1,3
         DO 50 J=1,3
*
C           ...FIND APPROPRIATE PART OF HSL AND REARRANGE TO A VECTOR...
*
            DO 55 K=1,NFS
            DO 55 L=1,NFS
               TEMP(K,L)=HSL(K+J-1,L-I+3)
55          CONTINUE
            DO 60 K=1,NFS
            DO 60 L=1,NFS
               RIH(L+(K-1)*8,1)=TEMP(L,K)
```

```
60          CONTINUE
*
C           ...FIND APPROPRITE XHAT...
*
            DX(1)=XFM(1)+(I-2)*DELXDY
            DX(2)=XFM(2)+(J-2)*DELYDY
            DX(3)=XFM(3)
            DX(4)=XFM(4)
            DX(5)=XFM(5)
            DX(6)=XFM(6)
            DX(7)=XFM(7)+(I-2)*DELXAT
            DX(8)=XFM(8)+(J-2)*DELYAT
*
C           ...MULTILY MEASUREMENT WITH XHAT...
*
            CALL MMPY(TEMP2,RIH,DX,NMS,ONE,NFS)
*
C           ...MULTIPLY BY AND SUM OVER NINE VALUES IN CPDF...
*
            DO 55 K=1,NMS
            DO 55 L=1,NFS
               HXHAT(K,L)=HXHAT(K,L)+TEMP2(K,L)*CPDF(I,J)
55          CONTINUE
50       CONTINUE
*
C        ...COMPUTE COEFFICIENT SCRIPT H, H...
*
*
C        ...REARRANGE HHAT TO A VECTOR...
*
         DO 70 K=1,NFS
         DO 70 L=1,NFS
            RIH(L+(K-1)*8,1)=HF(L,K)
70       CONTINUE
*
C        ...H =(HXHAT - (HHAT * XHAT)) * PFM ...

*
         CALL MMPY(H,RIH,XFM,NMS,ONE,NFS)
         CALL MADD(TEMP2,HXHAT,H,NMS,NFS,NFS)
         CALL MMPY(H,TEMP2,PFM,NMS,NFS,NFS)
*
         END
```

```
      SUBROUTINE INCHEK(ISPTL)

C          INCHEK ECHOS THE INPUT DATA AND PERFORMS SOME INITIAL
C     VARIABLE CHECKS.

C     ...VARIABLES...

      REAL IMAX
      CHARACTER ISPTL*3

      COMMON/IN/ NRUN,TFINAL,SIGS1,SIGAT,C(5),SIGF1,SIGF2,AR ,FIMAX ,
     :            SN,SIGMF0,KF,SIGMAB

      COMMON/FLIR/ XFOV,YFOV,NPIX,CSTH,SNTH,SIGMS,SIGFLR,ASPRO,
     :              IMEAS,VMAX,RANGE0,RANGE,IMAX,SIGVF,SIGPVF,AR

C     ...CALCULATE SIGNAL TO NOISE RATIO...
      SN=SIGMAB/IMAX
      IF(SN.LE.0.) SN=.001
      SN=1./SN

C     ...ECHO INPUT DATA...

      PRINT'(A)','1'
      PRINT'(T41,A//T49,A////T45,A)',
     :     'PROGRAM LEUTER-  A MONTE CARLO SIMULATION',
     :     'CAPT. LEUTHAUSER, GA-81D',
     :     '*** ******* INPUT DATA **********'

      PRINT'(/////T2,A,T20,A/T5,A)','NUMBER OF MONTE',
     :     'FINAL TIME','CARLO RUNS'
      PRINT'(/T6,I4,T22,F6.2)',NRUN,TFINAL

      PRINT'(//////T2,A)','TRUTH MODEL INPUTS..........'
      PRINT'(//T2,A,T24,A,T45,A,T59,A,T83,A)',
     :     '        DYNAMICS','   BACKGROUND NOISE',
     :     '      FLIR NOISE','      ATMOSPHERE',
     :     'PERPENDICULAR VELOCITY'
      PRINT'(T14,A,T32,A,T54,A,T75,A,T93,A)',
     :     'SIGMA','SIGMA','SIGMA','SIGMA','SIGMA'
      PRINT'(/T2,E17.10,T24,E17.10,T45,E17.10,T65,E17.10,T90,E17.10)',
     :     SIGS1,SIGMAB,SIGFLR,SIGAT,SIGMS

      PRINT'(//T2,A,T26,A,T51,A)','    MAXIMUM INTENSITY',
     :     '       ASPECT RATIO','SIGNAL TO NOISE RATIO'
      PRINT'(/T2,E17.10,T26,E17.10,T51,E17.10)',IMAX,ASPRO,SN

      IF (ISPTL .EQ. 'YES') THEN
         PRINT'(//T8,A)','SPATIAL NOISE COEFFICIENTS'
         PRINT'(/T5,A3,E13.6,T28,A3,E13.6,T5 ,A3,E13.6,T72,A3,E13.6,
     :         T94,A3,E13.6)','1= ',C(1),'2= ',C(2),'3= ',C(3),
     :         '4= ',C(4),'5= ',C(5)
      ENDIF

      PRINT'(//////T2,A)','FILTER INPUTS..........'
      PRINT'(//T2,A,T24,A,T45,A,T59,A)','        DYNAMICS',
     :     '      ATMOSPHERE','      MEASUREMENT','       VELOCITY'
```

```
      PRINT'(T10,A,T32,A,T54,A,T76,A)','SIGMA','SIGMA','SIGMA','SIGMA'
      PRINT'(/T2,E17.10,T24,E17.10,T46,E17.10,T68,E17.10)',
     :      SIGF1 ,SIGF2,RF,SIGMF

      PRINT'(//T2,A,T26,A)','     MAXIMUM INTENSITY','      ASPECT RATIO'
      PRINT'(/T2,E17.10,T26,E17.10)',FIMAXI,AR

      PRINT'(A)','1'
      END




      SUBROUTINE TRUTH (DT,SIGAT,SIGS1,PHI,BD,QD,Q)
C
C          TRUTH FORMULATES THE TRUTH MODEL STATE TRANSITION MATRIX,
C     PHI, THE DISCRETE INPUT MATRIX, BD, AND THE DISCRETE NOISE
C     STRENGTH MATRIX,QD.  DERIVATION OF THIS FORMULATION IS PRE-
C     SENTED IN THESES BY MERCIER AND BY HARNLY AND JENSEN.
C
      REAL DT,SIGAT,SIGS1,PHI(8,8),BD(8,2),QD(8,8),Q(3,3)
C
C     ...VARIABLE DEFINITIONS...
C
C     ATAU1- CONSTANT, ATMOSPHERIC JITTER SHAPING FILTER
C     ATAU2- CONSTANT, ATMOSPHERIC JITTER SHAPING FILTER
C     AGAIN- GAIN, ATMOSPHERIC JITTER SHAPING FILTER
C     SIGAT- TRUTH MODEL RMS ATMOSPHERIC ERROR, INPUT
C     SIGS1- TRUTH MODEL RMS DYNAMICS ERROR, INPUT
C     DT- DATA SAMPLE RATE, INPUT
C     PHI- STATE TRANSITION MATRIX, OUTPUT
C     BD- DISCRETE INPUT MATRIX, OUTPUT
C     QD- DISCRETE NOISE STRENGTH MATRIX, OUTPUT
C
C
C     ...DEFINE PARAMETERS...
C
      ATAU1=14.1
      ATAU2=583.3
      AGAIN = .3[illegible] 6534 * SIGAT
      DELT=-1.*DT
      FACT=(AGAIN**2)*(ATAU1**2)*(ATAU2**4)
      FACT1 = ATAU1-ATAU2
      FACT2 = ATAU1+ATAU2
      FACT3 = 2.*ATAU2
      B1 = FACT/(FACT1**4)
      B2 = FACT/(FACT1**3)
      B3 = FACT/(FACT1**2)
      R1 =  1.- EXP(2.*ATAU1*DELT)
      R2 =  1.- EXP(FACT2*DELT)
      R3 =  1.- EXP(2.*ATAU2*DELT)
      R4 = DT*EXP(DELT*FACT2)
      R5 = DT*EXP(2.*ATAU2*DELT)
```

```
C      ...ZERO OUT MATRICES...

       DO 10  I=1,8
          BD(I,1)=0.0
          BD(I,2)=0.0
          DO 10  J=1,8
             QD(I,J)=0.0
             PHI(I,J)=0.0
10     CONTINUE

C         FILL OUT TRUTH MODEL PHI MATRIX.
*
       PHI(1,1)= 1.
       PHI(2,2) = PHI(1,1)
       PHI(3,3) = EXP(ATAU1*DE_T)
       PHI(4,4) = EXP(ATAU2*DE_T)
       PHI(4,5) = DT*PHI(4,4)
       PHI(5,5) = PHI(4,4)
       PHI(6,6) = PHI(3,3)
       PHI(7,7) = PHI(4,4)
       PHI(7,8) = PHI(4,5)
       PHI(8,8) = PHI(5,5)

C         FILL OUT DISCRETE INPUT MATRIX
*
       BD(1,1)= DT
       BD(2,2)= DT

C         FILL THE QD MATRIX WITH VALUES USING EXACT INTEGRATION
*
       QD(1,1)= SIGS1
       QD(2,2) = QD(1,1)
       QD(3,3) = (G1*F1)/(2.*ATAU1)
       QD(3,4) = F2*(G2/FACT2**2-G1/FACT2)-R4*G2/FACT2
       QD(3,5) = G2*R2/FACT2
       QD(4,3) = QD(3,4)
       QD(4,4) = F3*(G1/FACT3-2.*G2/FACT3**2+2.*G3/FACT3**3)-
      * R3*(G2/ATAU2+G3*DT/FACT3-2.*G3/FACT3**2)
       QD(4,5) = R3*(G3/FACT3**2-G2/FACT3)-R5*G3/FACT3
       QD(5,3) = QD(3,5)
       QD(5,4) = QD(4,5)
       QD(5,5) = R3*G3/FACT3
       DO 20  I=3,5
       DO 20  J=3,5
          QD(I+3,J+3)=QD(I,J)
          Q(I-2,J-2)=QD(I,J)
20     CONTINUE
       END
      SUBROUTINE COVAR (PFP)
C           COVAR  INITIALIZES THE FILTER COVARIANCE MATRIX
C     FOR THE START OF EACH RUN.

      REAL PFP(8,8)

C     ...VARIABLE DEFINITIONS...

C     PFP- FILTER COVARIANCE AFTER UPDATE, OUTPUT


C     ...ZERO OUT MATRIX...

      DO 1  I=1,8
      DO 1   J=1,8
         PFP(I,J)= 0.
1     CONTINUE

C        FILL IN P+ AT TIME 0

      PFP(1,1)=25.
      PFP(2,2)=PFP(1,1)
      PFP(3,3)=2[illegible].
      PFP(4,4)=PFP(3,3)
      PFP(5,5)=4[illegible].
      PFP(6,6)=PFP(5,5)
      PFP(7,7)=.2
      PFP(8,8)=PFP(7,7)
      END
```

```
      SUBROUTINE FNOIZE(C,SIGMAB,RN,R)

C           FNOIZ  FORMULATES THE TRUTH MODEL BACKGROUND FLIR
C     SPATIAL NOISE MATRIX, RN.

      REAL C(5),SIGMAB,RN(64,64),R(64,64)


C     ...VARIABLE DEFINITION...

C     C- ARRAY OF SPATIAL NOISE COEFFICIENTS, INPUT
C     SIGMAB- TRUTH MODEL RMS BACKGROUND NOISE ERROR, INPUT
C     RN- SPATIAL NOISE CORRELATION COEFFICIENT MATRIX, OUTPUT
C     R- INTERNAL WORK AREA
```

```
C
C         SET UP SPATIAL NOISE CORRELATION COEFFICIENT MATRIX
C
      N=64
      M=8
      DO 36 I=1,N
      R(I,I)=1.
      IF (I.GE.64) GO TO 36
      R(I,I+1)=C(1)
      IF (I.GE.63) GO TO 36
      R(I,I+2)=C(3)
      IF (I.GE.55) GO TO 36
      R(I,I+6)=C(4)
      R(I,I+7)=C(2)
      R(I,I+8)=C(1)
      R(I,I+9)=C(2)
      R(I,I+10)=C(4)
      IF (I.GE.47) GO TO 36
      R(I,I+14)=C(5)
      R(I,I+15)=C(6)
      R(I,I+16)=C(3)
      R(I,I+17)=C(6)
      R(I,I+18)=C(5)
36    CONTINUE
      DO 37 I=1,M
      R(8*I-7,8*I)=0.0
      R(8*I-7,8*I-1)=0.0
      R(8*I-6,8*I)=0.0
      IF (I.GE.8) GO TO 37
      R(8*I,8*I+1)=0.0
      R(8*I,8*I+2)=0.0
      R(8*I-1,8*I+1)=0.0
      R(8*I-7,8*I+7)=0.0
      R(8*I-7,8*I+8)=0.0
      R(8*I-6,8*I+8)=0.0
      IF (I.GE.7) GO TO 37
      R(8*I,8*I+9)=0.0
      R(8*I,8*I+10)=0.0
      R(8*I-1,8*I+9)=0.0
      IF (I.GE.6) GO TO 37
      R(8*I,8*I+17)=0.0
      R(8*I,8*I+18)=0.0
      R(8*I-1,8*I+17)=0.0
37    CONTINUE
      DO 38 I=1,N
      L=I+1
      DO 38 J=L,N
      R(J,I)=R(I,J)
38    CONTINUE
      DO 39 I=1,N
      DO 39 J=1,N
      RN(I,J)=SIGMABAR*R(I,J)
39    CONTINUE
      END
      SUBROUTINE FILTR1(DT,PHIF,PHIFT)
C
C          FILTR  FORMULATES THE FILTER STATE TRANSITION MATRIX,
C     PHIF, AND ITS TRANSPOSE, PHIFT.
C
      REAL DT,PHIF(8,8),PHIFT(8,8)

C     ...VARIABLE DEFINITIONS...
C
C     ATAU1- CONSTANT, ATMOSPHERIC JITTER SHAPING FILTER
C     DT- DATA SAMPLE RATE, INPUT
C     PHIF- FILTER STATE TRANSITION MATRIX, OUTPUT
C     PHIFT- PHIF TRANSPOSE, OUTPUT
C

C     ...DEFINE PARAMETERS...
C
      ATAU1=14.1
      FTAU2=1. /ATAU1
      DELT=-1. *DT

C     ...ZERO OUT MATRIX...
      DO 10  I=1,8
      DO 10  J=1,8
         PHIF(I,J)=0.0
 10   CONTINUE

C         FILL OUT FILTER PHI MATRIX
C
      PHIF(1,1)=1.
      PHIF(1,3)=DT
      PHIF(1,5)=DT**2/2.
      PHIF(2,2)=PHIF(1,1)
      PHIF(2,4)=DT
      PHIF(2,6)=PHIF(1,5)
      PHIF(3,3)=PHIF(1,1)
      PHIF(3,5)=DT
      PHIF(4,4)=PHIF(1,1)
      PHIF(4,6)=DT
      PHIF(5,5)=1.
      PHIF(6,6)=1.
      PHIF(7,7)=EXP(DELT/FTAU2)
      PHIF(8,8)=PHIF(7,7)
C
C     FILL OUT PHIF TRANSPOSE
C
      DO 20  I=1,8
      DO 20  J=1,8
         PHIFT(I,J)=PHIF(J,I)
 20   CONTINUE
      END
      SUBROUTINE QAQUIR(DT,SIGF1 ,SIGF2,QFD)

C          QAQUIR FORMULATES THE ACQUISITION PHASE FILTER
C     DISCRETE NOISE STRENGTH MATRIX, QFD.

      REAL DT,SIGF1,SIGF2,QFD(8,8)

C     ...VARIABLE DEFINITIONS...

C     ATAU1- CONSTANT, ATMOSPHERIC JITTER SHAPING FILTER
C     DT- DATA SAMPLE RATE, INPUT
C     SIGF1- INITIAL FILTER RMS DYNAMICS ERROR, INPUT
C     SIGF2- FILTER RMS ATMOSPHERIC ERROR, INPUT
C     QFD- DISCRETE NOISE STRENGTH MATRIX, OUTPUT


C     ...DEFINE PARAMETERS...

      ATAU1=10.10
      FTAU2=1.0/ATAU1
      DELT=-1.0*DT
      SIGF1=ABS(SIGF1)

C     ...ZERO OUT MATRIX...

      DO 10 I=1,8
      DO 10 J=1,8
         QFD(I,J)=0.
10    CONTINUE

C     ...FILL OUT FILTER DISCRETE QFD MATRIX
C        FOR START OF ACQUISITION PHASE...

      QFD(1,1)=DT**5*SIGF1/20.
      QFD(1,3)=DT**4*SIGF1/8.
      QFD(1,5)=DT**3*SIGF1/6.
      QFD(2,2)=QFD(1,1)
      QFD(2,4)=QFD(1,3)
      QFD(2,6)=QFD(1,5)
      QFD(3,1)=QFD(1,3)
      QFD(3,3)=DT**3*SIGF1/3.
      QFD(3,5)=DT**2*SIGF1/2.
      QFD(4,2)=QFD(3,1)
      QFD(4,4)=QFD(3,3)
      QFD(4,6)=QFD(3,5)
      QFD(5,1)=QFD(1,5)
      QFD(5,3)=QFD(3,5)
      QFD(5,5)=SIGF1*DT
      QFD(6,2)=QFD(1,5)
      QFD(6,4)=QFD(3,5)
      QFD(6,6)=QFD(5,5)
      QFD (7,7) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD (8,8)=QFD (7,7)
      END
      SUBROUTINE TRUTH(XCENTR,YCENTR,NPS,ONE)
C          TRUTH PERFORMS THE TRUTH MODEL SIMULATION FOR EACH STEP.
C
C     ...VARIABLES...

      REAL XCENTR,YCENTR
      INTEGER NPS,ONE

      COMMON/ARRAY/ XFP(8),XFPO(8),XFM(8),PFP(8,8),PFPOLD(8,8),
     !              PFFP(8,8),PFM(8,8),QFD(8,8),EXTRA(8,8),PHIF(8,8),
     !              PHIFT(8,8),UT(2,1),PL2P(64,2),BD( ,2),TEMP(8,8),
     !              TEMP1(8,8),SQQD(8,8),W(8),XS( ),H(64,8),HF(8,8),
     !              Z(8,8),R(64,64),WKAREA(64,64),HT(8,64),PHI(8,8)

C     ...PERFORM TRUTH MODEL SIMULATION...

      XCENTR=XCENTR+UT(1,1)*BD(1,1)
      YCENTR=YCENTR+UT(2,1)*BD(2,2)
      CALL NOISE (NPS,W)
      CALL MMPY(TEMP,SQQD,W,NPS,NPS,ONE)
      CALL MMPY(TEMP1,PHI,XS,NPS,NPS,ONE)
      CALL MADD(XS,TEMP,TEMP1,NPS,ONE,ONE)
      CALL MMPY(TEMP1,BD,UT,NPS,2,ONE)
      CALL MADD(XS,XS,TEMP1,NPS,ONE,ONE)
      END

      SUBROUTINE MOVEUP(NFS,NFS2,ONE)

C          MOVEUP PROPAGATES BOTH THE FILTER STATE VECTOR AND IT'S
C     COVARIANCE MATRIX FROM ONE UPDATE TO THE NEXT.
C
C     ...VARIABLES...

      INTEGER NFS,NFS2,ONE

      COMMON/ARRAY/ XFP(8),XFPO(8),XFM(8),PFP(8,8),PFPOLD(8,8),
     !              PFFP(8,8),PFM(8,8),QFD(8,8),EXTRA(8,8),PHIF(8,8),
     !              PHIFT(8,8),UT(2,1),PL2P(64,2),BD(8,2),TEMP(8,8),
     !              TEMP1(8,8),SQQD(8,8),W(8),XS( ),H(64,8),HF(8,8),
     !              Z(8,8),R(64,64),WKAREA(64,64),HT( ,64),PHI(8,8)

C     ...FILTER STATE PROPAGATION...

      CALL SHIFTA(XFF,XFPO,NFS,NFS)
      CALL MMPY(XFM,PHIF,XFP,NFS,NFS,ONE)

C     ...FILTER COVARIANCE PROPAGATION...

      CALL SHIFTA(PFF,PFPOLD,NFS2,NFS2)
      CALL MMPY(EXTRA,PHIF,PFP,NFS,NFS,NFS)
      CALL MMPY(PFFF,EXTRA,PHIFT,NFS,NFS,NFS)
C        PPFP= PHI * P(TI-1)+ * PHIT
      CALL MADD(PFM,PPFP,QFD,NFS,NFS,ONE)
C        PFM = P(TI-)=PHI * P(TI-1)+ * PHIT + G   Q   GT
      END
      SUBROUTINE XFORM(TIME,ALFADT,BETADT,XVEH,YVEH,ZVEH,VMAX,RANGE)
C
C          XFORM TRANSFORMS THE VEHICLE'S POSITION FROM THE 3-D
C     REFERENCE FRAME TO THE 2-D FLIR IMAGE PLANE.
*
      REAL TIME,ALFADT,BETADT,XVEH,YVEH,ZVEH,XDOT,YDOT,ZDOT,RANGE,VMAX
*
C     ...VARIABLE DEFINITIONS...
*
C     TIME- CURRENT TIME IN THE SIMULATION, INPUT
C     ALFADT- TRUE AZIMUTH RATE, OUTPUT (PIXELS/SECOND)
C     BETADT- TRUE ELEVATION RATE, OUPUT (PIXELS/SECOND)
C     X,Y,ZVEH- VEHICLE POSITION FROM TRAJEC, OUTPUT (METERS)
C     X,Y,ZDOT- VEHICLE VELOCITY FROM TRAJEC (METERS/SECOND)
C     VMAX-                                  , OUTPUT (PIXELS/SEC)
C     RANGE- VEHICLE'S RANGE FROM THE TRACKER, OUTPUT (METERS)
*
C     ...PERFORM TRANSFORMATION...
*
      CALL TRAJEC(TIME,XVEH,YVEH,ZVEH,XDOT,YDOT,ZDOT)
      RHOR=(XVEH*XVEH)+(ZVEH*ZVEH)
      RANGE=RHOR+(YVEH*YVEH)
      ALFADT=(ZVEH*XDOT-XVEH*ZDOT)/(RHOR*[illegible] 2)
      RHOR=SQRT(RHOR)
      BETADT=(RHOR*YDOT-(YVEH*((XVEH*XDOT+ZVEH*ZDOT)/RHOR)))/
     :(RANGE*[illegible] 2)
      RANGE=SQRT(RANGE)
      VMAX=SQRT(XDOT*XDOT+YDOT*YDOT+ZDOT*ZDOT)/(RANGE*[illegible] 2)
*
      END




      SUBROUTINE TRAJEC(TIME,XVEH,YVEH,ZVEH,XDOT,YDOT,ZDOT)
*
C          TRAJEC DEFINES THE VEHICLE'S TRAJECTORY FOR THE
C     TRUTH MODEL.  THE TRAJECTORY IS DEFINED IN A 3-D REFERENCE
C     FRAME WHOSE ORIGIN IS AT THE TRACKER.
*
      REAL TIME,XVEH,YVEH,ZVEH,XDOT,YDOT,ZDOT
      REAL DT2,OMEGA,RADIUS
*
C     ...VARIABLE DEFINITIONS...
*
C     TIME- CURRENT TIME IN THE SIMULATION, INPUT (SECONDS)
C     XVEH- VEHICLE POSITION IN THE X-DIRECTION, OUTPUT (METERS)
C     YVEH- VEHICLE POSITION IN THE Y-DIRECTION, OUTPUT (METERS)
C     ZVEH- VEHICLE POSITION IN THE Z-DIRECTION, OUTPUT (METERS)
C     XDOT- VEHICLE VELOCITY IN THE X-DIRECTION, OUTPUT (METERS/SEC)
C     YDOT- VEHICLE VELOCITY IN THE Y-DIRECTION, OUTPUT (METERS/SEC)
C     ZDOT- VEHICLE VELOCITY IN THE Z-DIRECTION, OUTPUT (METERS/SEC)
C     DT2- SAMPLE RATE/2
*
```

```
C       ...DEFINE PARAMETERS...
*
        DT2 = 1. /60.0
*
C       ...DEFINE TRAJECTORY...
*
C                CONSTANT VELOCITY CROSS RANGE
*
        XDOT=-530.33
        YDOT=-53[illegible].33
        ZDOT=0.0
        XVEH=1[illegible]0[illegible].0+(XDOT*TIME)
        YVEH=180[illegible].0+(YDOT*TIME)
        ZVEH=150 0.0
        END



C       ...DEFINE PARAMETERS...

        DT2 = 1. /[illegible] .
C                 1.G TURN
        OMEGA =  .13[illegible]7[illegible]33
        RADIUS = [illegible]3[illegible].[illegible]37
*
C       ...DEFINE TRAJECTORY...
*
        IF (TIME .LE. 2.[illegible]) THEN
        XDOT=-7[illegible] .
        YDOT= .[illegible]
        ZDOT=[illegible].[illegible]
        XVEH=7[illegible] . +(XDOT TIME)
        YVEH=[illegible] .
        ZVEH=1[illegible] .[illegible]
*
        ELSE
C            BEGIN TURN
*
        XDOT=-7[illegible] .*COS(OMEGA*(TIME-DT2-2.[illegible]))
        YDOT=[illegible] .*SIN(OMEGA*(TIME-DT2-2. ))
        ZDOT=[illegible].
        XVEH=[illegible] . -RADIUS*(SIN(OMEGA*(TIME-2. )))
        YVEH=[illegible] .[illegible]+RADIUS*(1.[illegible]-COS(OMEGA*(TIME-2.[illegible])))
        ZVEH=1[illegible] .[illegible]
        ENDIF
        END
```

```
      SUBROUTINE QTUNE(QFD,TIME,DT,SIGF1)
*
C          QTUNE PROVIDES TUNING CAPABILITY FOR THE FILTER
C     DISCRETE NOISE STRENGTH MATRIX, QFD.
*
      REAL QFD(8,8),TIME,DT,SIGF1,VARYQ
      SAVE VARYQ
*
C     ....ACQUISITION VARYQ....
*
      IF (TIME .LT. 0.04) VARYQ=SIGF1
      IF (TIME .LT. 0.5) VARYQ=VARYQ-(.05*SIGF1)
*
C     ....FILL OUT FILTER DISCRETE QFD MATRIX....
*
      QFD(1,1)=DT**5*VARYQ/20.
      QFD(1,3)=DT**4*VARYQ/8.
      QFD(1,5)=DT**3*VARYQ/6.
      QFD(2,2)=QFD(1,1)
      QFD(2,4)=QFD(1,3)
      QFD(2,6)=QFD(1,5)
      QFD(3,1)=QFD(1,3)
      QFD(3,3)=DT**3*VARYQ/3.
      QFD(3,5)=DT**2*VARYQ/2.
      QFD(4,2)=QFD(3,1)
      QFD(4,4)=QFD(3,3)
      QFD(4,6)=QFD(3,5)
      QFD(5,1)=QFD(1,5)
      QFD(5,3)=QFD(3,5)
      QFD(5,5)=VARYQ*DT
      QFD(6,2)=QFD(1,5)
      QFD(6,4)=QFD(3,5)
      QFD(6,6)=QFD(5,5)
*
      END
```

```
      SUBROUTINE NOISE(N,W)
      DIMENSION W(N)
      DO 15 J=1,N
      TOTAL=0.
      DO 5 I=1,12
      TOTAL =TOTAL + RANF()
    5 CONTINUE
      W(J) = TOTAL - 6.
   15 CONTINUE
      RETURN
      END
      SUBROUTINE CHOLEX(A,S,N)
      DIMENSION A(1),S(1)
      NDIM=N
      NDIM1=N+1
      TOL=1.E-5
      MR=0
      NN=N*NDIM
      TOL1=0.
      DO 1 I=1,NN,NDIM1
      R=ABS(A(I))
1     IF(R.GT.TOL1)TOL1=R
      TOL1 = TOL1*1.E-12
      II=1
      DO 5  I=1,N
      IM1 = I-1
      DO 5 JJ=I,NN,NDIM
5     S(JJ) = 0.
      ID = II+IM1
      R=A(ID)-DOT(IM1,S(II),S(II))
      IF(ABS(R).LT.(TOL*A(ID)+TOL1)) GO TO 5
      IF(R) 15,5 ,20
15    MR=-1
      PRINT*,'CHOLESK TRIED TO FACTOR AN INDEFINITE MATRIX'
      RETURN
20    S(ID) = SQRT(R)
      MR=MR+1
      IF(I.EQ.N) RETURN
      L=II+NDIM
      DO 25 JJ=L,NN,NDIM
      IJ=JJ+IM1
25    S(IJ) = (A(IJ)-DOT(IM1,S(II),S(JJ)))/S(ID)
30    II=II+NDIM
      RETURN
      END
```

```
      SUBROUTINE MMPY(C,A,B,K,M,N)
      DIMENSION C(K,N),A(K,M),B(M,N)
      DO 1 I=1,K
      DO 1 J=1,N
      C(I,J)= 0.
    1 CONTINUE
      DO 5 L=1,K
      DO 5 J=1,N
      DO 5 I=1,M
      C(L,J) = C(L,J) + (A(L,I)*B(I,J))
    5 CONTINUE
      RETURN
      END
      SUBROUTINE MADD(C,A,B,J,K,IFLAG)
      DIMENSION A(J,K),B(J,K),C(J,K)
      IF(IFLAG.EQ.1) GO TO 6
      DO 5 N=1,J
      DO 5 M=1,K
      C(N,M) = A(N,M) - B(N,M)
    5 CONTINUE
      RETURN
    6 DO 10  N=1,J
      DO 10  M=1,K
      C(N,M) = A(N,M) + B(N,M)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MWRITE(A)
      REAL A(64,64)
      IBLOCK=1
      DO 10  IB=1,57,8
         DO 10  IA=1,57,8
            PRINT '(/I2,A,I2/)','BLOCK ',IBLOCK
            IBLOCK=IBLOCK+1
            PRINT '(8(8(2X,G13.7)/))',((A(I,J),J=IA,IA+7),I=IB,IB+7)
10    CONTINUE
      END
```

```
      SUBROUTINE MOUT(A)
      REAL A(8,8)
      PRINT '(8(8(2X,G13.7)/))',((A(I,J),J=1,8),I=1,8)
      END
```

```
      SUBROUTINE MOUT1(A)
      REAL A(8,2)
      PRINT '(8(2(2X,G13.7)/))',((A(I,J),J=1,2),I=1,8)
      END
```

```
      SUBROUTINE MOUT2(A)
      REAL A(8)
      PRINT '(8(2X,G13.7))',(A(I),I=1,8)
      END
```

```
      SUBROUTINE MOUT3(A)
      REAL A(8,64)
      PRINT '(64 (8(2X,G13.7)/))',((A(I,J),I=1,8),J=1,64)
      END
      SUBROUTINE SHIFT(A,B,IS,N)
      DIMENSION A(N),B(N)
C        A=INPUT ARRAY,B=OUTPUT ARRAYN=ARRAY SIZE,IS=AMT OF SHIFT
C
      NN=N
      DO 300 I=1,N
  300 B(I)=0.
C        TEST FOR LEFT OR RIGHT SHIFT
      IF(IS.LE.0) GO TO 100
C        EXECUTE RIGHT SHIFT
      MM=1+IS
      M=N-IS
      CALL SHIFTA(A(1),B(MM),M,NN)
      GO TO 200
C        EXECUTE LEFT SHIFT
  100 MM=1-IS
      M=N+IS
      CALL SHIFTA(A(MM),B(1),M,NN)
  200 RETURN
      END



      SUBROUTINE SHIFTA(A,B,M,N)
      DIMENSION A(N),B(N)
      DO 100 K=1,M
  100 B(K) = A(K)
      RETURN
      END



      SUBROUTINE RNDF(A,K)
C        SUBROUTINE TO ROUNDOFF A,TO NEAREST INTEGER,K
C
      K=IFIX(A)
      B=A-K
      K=K+1
      IF(B.LT.0.5)GO TO 100
  100 RETURN
      END



      FUNCTION DOT(NR,A,B)
      DIMENSION A(1),B(1)
      DOT=0.
      DO 1 I=1,NR
    1 DOT = DOT+A(I)*B(I)
      RETURN
      END
      PROGRAM PLOTTER(OUTPUT,TAPE8,PLOT,TAPE6=OUTPUT)
      REAL KALCORR
      COMMON XM(152),VAR(152),EVAR(152),XMPVAR(152),XMMVAR(152)
      COMMON/MATRIX/KALCORR(14,20,150),STATS(4,3,150),ERROR(2,2,150),
     * ZERO(152),TYME(152),VPT(4,4,21)
      REWIND 8
      NFS=4
      DT=1./30.
      REWIND 8
      NFILE=8
      NTIME=150
      NRUN=2
      CALL PROCESS(NFS,NRUN,NTIME,DT,NFILE)
      CALL OUTPUT(NFS,3,NTIME,5,DT)
      CALL GRAPH(NTIME,NRUN)
      STOP
      END
```

```
      SUBROUTINE PROCESS(NFS,NRUN,NTIME,DT,NFILE)
      REAL KALCORR
      COMMON/MATRIX/KALCORR(14,20,150),STATS(4,3,150),ERROR(2,2,150),
     *  ZERO(152),TYME(152),VPT(4,4,21)
      DIMENSION ESUM(5),SUMSQ(6),RMSUM(4)
      DO 51  I=1,NRUN
      DO 51  J=1,NTIME
      READ(NFILE) (KALCORR(L,I,J),L=1,14)
   51 CONTINUE
      R=FLOAT(NRUN)
      R1 = FLOAT(NRUN-1)
      DO 30 I=1,NTIME
      ZERO(I) = 0.
      TYME(I) = FLOAT(I)*DT
      DO 1 K=1,6
      ESUM(K) = 0.
      SUMSQ(K) = 0.
    1 CONTINUE
      DO 2 K=1,4
      RMSUM(K) = 0.
    2 CONTINUE
      DO 5 L=1,4
      DO 5 J=1,NRUN
      ERR = KALCORR(L,J,I) - KALCORR(L+4,J,I)
      ESUM(L) = ESUM(L) + ERR
      SUMSQ(L) = SUMSQ(L) + ERR**2
      RMSUM(L) = RMSUM(L) + KALCORR(L+10,J,I)
    5 CONTINUE
```

```
      DO 10  J=1,NRUN
      ERR1 = KALCORR(1,J,I) - KALCORR(9,J,I)
      ERR2 = KALCORR(3,J,I) - KALCORR(10,J,I)
      ESUM(5) = ESUM(5) + ERR1
      ESUM(6) = ESUM(6) + ERR2
      SUMSQ(5) = SUMSQ(5) + ERR1**2
      SUMSQ(6) = SUMSQ(6) + ERR2**2
   10 CONTINUE
      DO 20 K=1,4
      EMEAN = ESUM(K) /R
      ESTDEV = SQRT((SUMSQ(K)-R*EMEAN**2)/RM)
      FSTDEV = SQRT(RMSUM(K)/R)
      STATS(K,1,I) = EMEAN
      STATS(K,2,I) = ESTDEV
      STATS(K,3,I) = FSTDEV
   20 CONTINUE
      DO 21 K=1,2
      EMEAN = ESUM(K+4)/R
      ESTDEV = SQRT((SUMSQ(K+4)-R*EMEAN**2)/RM)
      ERROR(K,1,I) = EMEAN
      ERROR(K,2,I) = ESTDEV
   21 CONTINUE
   99 CONTINUE
      DO 55 L=1,4
      DO 51 I=1,6
      ESUM(I)=KALCORR(L,1,I*30)-KALCORR(L+4,1,I*30)
      SUMSQ(I)=ESUM(I)**2
 51   CONTINUE
      VPT(L,1,1)= VPT(L,2,1)= VPT(L,3,1)= VPT(L,4,1)= 0.
      DO 55 I=2,20
      R=FLOAT(I)
      RM=FLOAT(I-1)
      DO 60  J=1,4
      ITIME=J*30
      ERR=KALCORR(L,J,ITIME)-KALCORR(L+4,I,ITIME)
      ESUM(J)= ESUM(J)+ERR
      SUMSQ(J)= SUMSQ(J)+ERR**2
      EMEAN= ESUM(J)/R
      ESTDEV= SQRT((SUMSQ(J)-R*EMEAN**2)/RM)
      VPT(L,J,I)= ESTDEV
 60   CONTINUE
 55   CONTINUE
 65   CONTINUE
      RETURN
      END
      SUBROUTINE OUTPUT(NES,NSTAT,NTIME,NFILE,DT)
      REAL KALCORR
      COMMON/MATRIX/KALCORR(14,2 ,150),STATS(4,3,150),ERROR(2,2,150 ),
     * ZERO(152),TYME(152),VPT(4,4,21)
      DO 50  I=1,NES
      TIME=0.
      WRITE(NFILE,1) I
    1 FORMAT(//2X,"********** THE FOLLOWING IS OUTPUT FOR ERROR STATE ",
     *I1,2X,"**********",//)
      WRITE(NFILE,2)
    2 FORMAT(2X,"TIME",5X,"MEAN ERROR",5X,"STD OF ERROR",5X,"FILTER EST"
     *,"IMATE OF STD"/)
      DO 25  J=1,NTIME
      TIME = TIME+DT
      WRITE(NFILE,3)TIME,(STATS(I,K,J),K=1,NSTAT)
    3 FORMAT(1X,F5.  ,3(5X,E12.5))
   25 CONTINUE
   50 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GRAPH(NTIME,NRUN)
      REAL KALCORR
      COMMON XM(152),VAR(152),FVAR(152),XMPVAR(152),XMMVAR(152)
      COMMON/MATRIX/KALCORR(14,2 ,150),STATS(4,3,150),ERROR(2,2,150),
     * ZERO(152),TYME(152),VPT(4,4,21)
      DIMENSION AD(17),BD(17),CD(17),DD(17),FD(17),ED(17),GD(17),HD(17)
     *,QD(17),RD(17),SD(17),TD(17)
      DATA AD(1)/20HMEAN ERROR +-1 SIGMA/
      DATA AD(3)/20H  TARGET DYNAMICS   /
      DATA AD(5)/20H      X CHANNEL     /
      DATA AD(7)/20H       FLIR FOV     /
      DATA AD(9)/20H      TIME (SEC)    /
      DATA AD(11)/20H         PIXELS      /
      DATA AD(13)/50H               Y CHANNEL DYNAMICS ERROR (S/N=    )       /
      BD(1)=CD(1)=DD(1)=AD(1)
      BD(2)=CD(2)=DD(2)=AD(2)
      BD(7)=CD(7)=DD(7)=ED(7)=FD(7)=GD(7)=HD(7)=AD(7)
      BD(8)=CD(8)=DD(8)=ED(8)=FD(8)=GD(8)=HD(8)=AD(8)
      BD(9)=CD(9)=DD(9)=ED(9)=FD(9)=GD(9)=HD(9)=QD(9)=RD(9)=SD(9)=TD(9)=
     *AD(9)
      BD(10)=CD(10)=DD(10)=ED(10)=FD(10)=GD(10)=HD(10)=QD(10)=RD(10)=
     *SD(10)=TD(10)=AD(10)
      BD(11)=CD(11)=DD(11)=ED(11)=FD(11)=GD(11)=HD(11)=QD(11)=RD(11)=
     *SD(11)=TD(11)=AD(11)
      BD(12)=CD(12)=DD(12)=ED(12)=FD(12)=GD(12)=HD(12)=QD(12)=RD(12)=
     *SD(12)=TD(12)=AD(12)
      BD(5)=FD(5)=GD(5)=QD(5)=RD(5)=AD(5)
      BD(6)=FD(6)=GD(6)=QD(6)=RD(6)=AD(6)
      DATA CD(3)/20H      Y CHANNEL       /
      DD(5)=FD(5)=HD(5)=SD(5)=TD(5)=CD(5)
      DD(6)=FD(6)=HD(6)=SD(6)=TD(6)=CD(5)
      DATA BD(3)/20H        VELOCITY      /
      DD(7)=AD(7)=TD(7)=BD(3)
      DD(8)=BD(8)=TD(8)=BD(4)
      CD(3)=QD(7)=SD(7)=AD(3)
      CD(4)=QD(8)=SD(8)=AD(4)
      DATA RD(13)/50H              X CHANNEL VELOCITY ERROR (S/N=    )       /
      DATA QD(1)/20H ACTUAL VS. FILTER  /
      DATA QD(3)/20H       SIGMA        /
      RD(1)  =  SD(1)  =  TD(1)  =  QD(1)
      RD(2)  =  SD(2)  =  TD(2)  =  QD(2)
      RD(3)  =  SD(3)  =  TD(3)  =  QD(3)
      RD(4)  =  SD(4)  =  TD(4)  =  QD(4)
      DATA CD(13)/50H              Y CHANNEL DYNAMICS ERROR (S/N=  )         /
      DATA DD(13)/50H              Y CHANNEL VELOCITY ERROR (S/N=    )       /
      DATA QD(13)/50H              FILTER VS. ACTUAL SIGMA PLOT  (S/N =    )/
      DATA ED(1)/20HMEAN TRACKING ERROR /
      DATA ED(3)/20H  FOR FILTER MODEL  /
      SD(13)=TD(13)=RD(13)=QD(13)
      SD(14)=TD(14)=RD(14)=QD(14)
      SD(15)=TD(15)=RD(15)=QD(15)
      SD(16)=TD(16)=RD(16)=QD(16)
      SD(17)=TD(17)=RD(17)=QD(17)
      DATA GD(3)/20HFOR CORRELATOR MODEL/
      FD(1)  =  GD(1)  =  HD(1)  =  ED(1)
      FD(2)  =  GD(2)  =  HD(2)  =  ED(2)
      FD(3)=ED(3)
      FD(4)=ED(4)
      HD(3)=GD(3)
      HD(4)=GD(4)
      DATA ED(13)/50H          X CHANNEL FILTER TRACKING ERROR (S/N =    )/
      DATA FD(13)/50H          Y CHANNEL FILTER TRACKING ERROR (S/N =    )/
      DATA GD(13)/50H      X CHANNEL CORRELATOR TRACKING ERROR (S/N =    )/
      DATA HD(13)/50H      Y CHANNEL CORRELATOR TRACKING ERROR (S/N =    )/
      CALL PLOTS(3., ,ALPLOT)
      CALL PLOT(1.85,-2.1,-3)
      CALL SCALE(TYME,7.,150,1)
      DO 2  M=1,3,2
      DO 5 I=1,NTIME
      XM(I)  =  STATS(M,1,I)
      XMPVAR(I)  =  STATS(M,1,I)  + STATS(M,2,I)
      XMMVAR(I)  =  STATS(M,1,I)  - STATS(M,2,I)
      FVAR(I)  =  STATS(M,3,I)
      VAR(I)  =  STATS(M,2,I)
    5 CONTINUE
      CALL PLOT(1.75,-2.1,-3)
      GO TO  (6,8),M
    6 CALL PTNT(AD,QD,1)
      GO TO 20
    8 CALL PTNT(CD,SD,1)
      GO TO 20
   20 CONTINUE
      CALL PLOT(1.75,-2.1,-3)
      CALL PLOTE(F)
      RETURN
      END
      SUBROUTINE FINT(A,B,IFLAG)
      REAL KALCOEF
      COMMON XM(152),VAR(152),FVAR(152),XMPVAR(1 2),XMMVAR(152)
      COMMON/MATRIX/KALCORR(14,2 ,150),STATS( ,3,15 ),ERROR(2,2,15 ),
     * ZERO(152),TYME(152),VPT(4,4,21)
      DIMENSION A(17),B(17)
      VAR(151)=FVAR(151)=XMPVAR(151)=XMMVAR(151)= .
      VAR(152)=FVAR(152)=XMPVAR(152)=XMMVAR(152)= .
      CALL SCALE(XMPVAR,4.5,302,1)
      XMPVAR(151)=XM(151)=ZERO(151)=XMMVAR(151)
      XMPVAR(152)=XM(152)=ZERO(152)=XMMVAR(152)=-XMMVAR(152)
      CALL VGRAPH(TYME,XM,15 ,A,-1,15,4)
      CALL VGRAPH(TYME,XMPVAR,15 ,A,2,15,14)
      CALL VGRAPH(TYME,XMMVAR,15 ,A,2,15,14)
      IF(IFLAG.EQ. )GO TO 99
      CALL SCALE(VAR,4.5,302,1)
      VAR(151)=FVAR(151)
      VAR(152)=FVAR(152)=-FVAR(152)
      CALL VGRAPH(TYME,VAR,15 ,B,-2,15,4)
      CALL VGRAPH(TYME,FVAR,15 ,B,2,15,4)
   99 RETURN
      END
```

```
      SUBROUTINE VGRAPH(X,Y,N,ID,NO,NP,NS)
      REAL KALCOEF
      COMMON XM(152),VAR(152),FVAR(152),XMPVAR(1 2),XMMVAR(152)
      COMMON/MATRIX/KALCORR(14,2 ,15 ),STATS( ,3,15 ),ERROR(2,2,15 ),
     * ZERO(152),TYME(152),VPT( , ,21)
      DIMENSION X(152),Y(152),ID(17)
      IF(NO.EQ.2) GO TO 3
      CALL PLOT(1.55,-2.1,-3)
      CALL PLOT( .,.1,2)
      CALL PLOT( .,1 .9,3)
      CALL PLOT( .,11.,2)
      CALL PLOT( . ,11.,3)
      CALL PLOT( . ,1 .9,2)
      CALL PLOT( .5,.1,3)
      CALL PLOT( .5, .,2)
      CALL PLOT( ., ., )
      CALL PLOT(1.35,1.35,3)
      CALL PLOT(1.35,9.65,2)
      CALL PLOT(1.45,9.55,3)
      CALL PLOT(1. 5,7.55,2)
      DO 2  I=1, ,2
      CALL SYMBOL((1.45+(I+1.5)*. ),7. 5,. ,ID(I), .,21)
2     CONTINUE
```

```
      CALL PLOT(1.45,7.55,3)
      CALL PLOT(2.45,7.55,2)
      CALL PLOT(2.45,9.55,2)
      CALL PLOT(1.45,9.55,2)
      CALL PLOT(1.35,9.65,3)
      CALL PLOT(7.45,9.65,2)
      CALL PLOT(7.45,1.35,2)
      CALL PLOT(1.35,1.35,2)
      CALL SYMBOL(1.85,1.115,.105,ID(13),0.,50)
      CALL PLOT(8.95,2.1,-3)
      Y2=-Y(N+2)
      Y1= Y(N+1)
      X1= X(N+1)
      X2= X(N+2)
      CALL AXIS(0.,0.,ID(9),-20,7.,30.,X1,X2)
      CALL AXIS(0.,0.,ID(11),20,4.5,130.,Y1,Y2)
      IF(NP.EQ.-1)GO TO 30
      CALL LINE(ZERO,TYME,N,1,0,NS)
   30 CONTINUE
      CALL LINE(Y,X,N,1,NP,NS)
      RETURN
      END
```

```
      SUBROUTINE VARFT(ID,N,L)
      REAL KALCORR
      COMMON XM(152),VAR(152),FVAR(152),XMPVAR(152),YMMVAR(152)
      COMMON/MATRIX/KALCORR(10,20,150),STATS(5,3,150),ERROR(2,2,150),
     * ZERO(152),TYME(152),VPT(4,4,21)
      DIMENSION Y(22),ID(17),RUNS(72),DUM(84),VPLOT(4,21)
      EQUIVALENCE(DUM(1),VPLOT(1,1))
      DO 10 I=1,4
      DO 10 J=1,21
      VPLOT(I,J)= VPT(L,I,J)
   10 CONTINUE
      CALL PLOT(1.85,-2.1,-3)
      CALL PLOT(0.,.1,2)
      CALL PLOT(0.,10.9,3)
      CALL PLOT(0.,11.,2)
      CALL PLOT(8.5,11.,3)
      CALL PLOT(8.5,10.9,2)
      CALL PLOT(8.5,.1,3)
      CALL PLOT(8.5,0.,2)
      CALL PLOT(0.,0.,3)
      CALL PLOT(1.35,1.35,3)
      CALL PLOT(1.35,9.65,2)
      CALL PLOT(1.45,9.35,3)
      CALL PLOT(1.45,7.35,2)
      ID(1)=10H      VARI
      ID(2)=10HANCE
      DO 20 I=1,3,2
      CALL SYMBOL((1.45+(I+1.5)*.1),7.55,.08,ID(I),0.,20)
   20 CONTINUE
```

```
      ED(1)=26H      MEAN ERROR
      CALL PLOT(1.45,7.35,3)
      CALL PLOT(2.45,7.35,2)
      CALL PLOT(2.45,9.35,2)
      CALL PLOT(1.45,9.35,2)
      CALL PLOT(1.35,9.65,3)
      CALL PLOT(7.45,9.65,2)
      CALL PLOT(7.45,1.35,2)
      CALL PLOT(1.35,1.35,2)
      CALL SYMBOL(1.85,1.115,.105,
     X38H                    VARIANCE CONVERGENCE              ,0.,38)
      CALL PLOT(0.95,2.1,-3)
      CALL SCALE(DUM,4.5,80,1)
      DO 3  I=1,N
      RUNS(I)=FLOAT(I)
3     CONTINUE
      RUNS(N+1)=1$RUNS(N+2)=3
      CALL AXIS(0.,0.,8HVARIANCE,8,4.5,180.,DUM(81),DUM(82))
      CALL AXIS(0.,0.,3HRUN,-3,7.,0.,1.,3.)
      Y(N+1)= DUM(81)
      Y(N+2)=-DUM(82)
      DO 4  J=1,4
      DO 5  I=1,N
      Y(I)= VPLOT(J,I)
5     CONTINUE
      CALL LINE(Y,RUNS,N,1,1,J)
4     CONTINUE
      RETURN
      END
```

Appendix E

Appendix E

Performance Plots for the Extended Kalman Filter

Constant Parameters

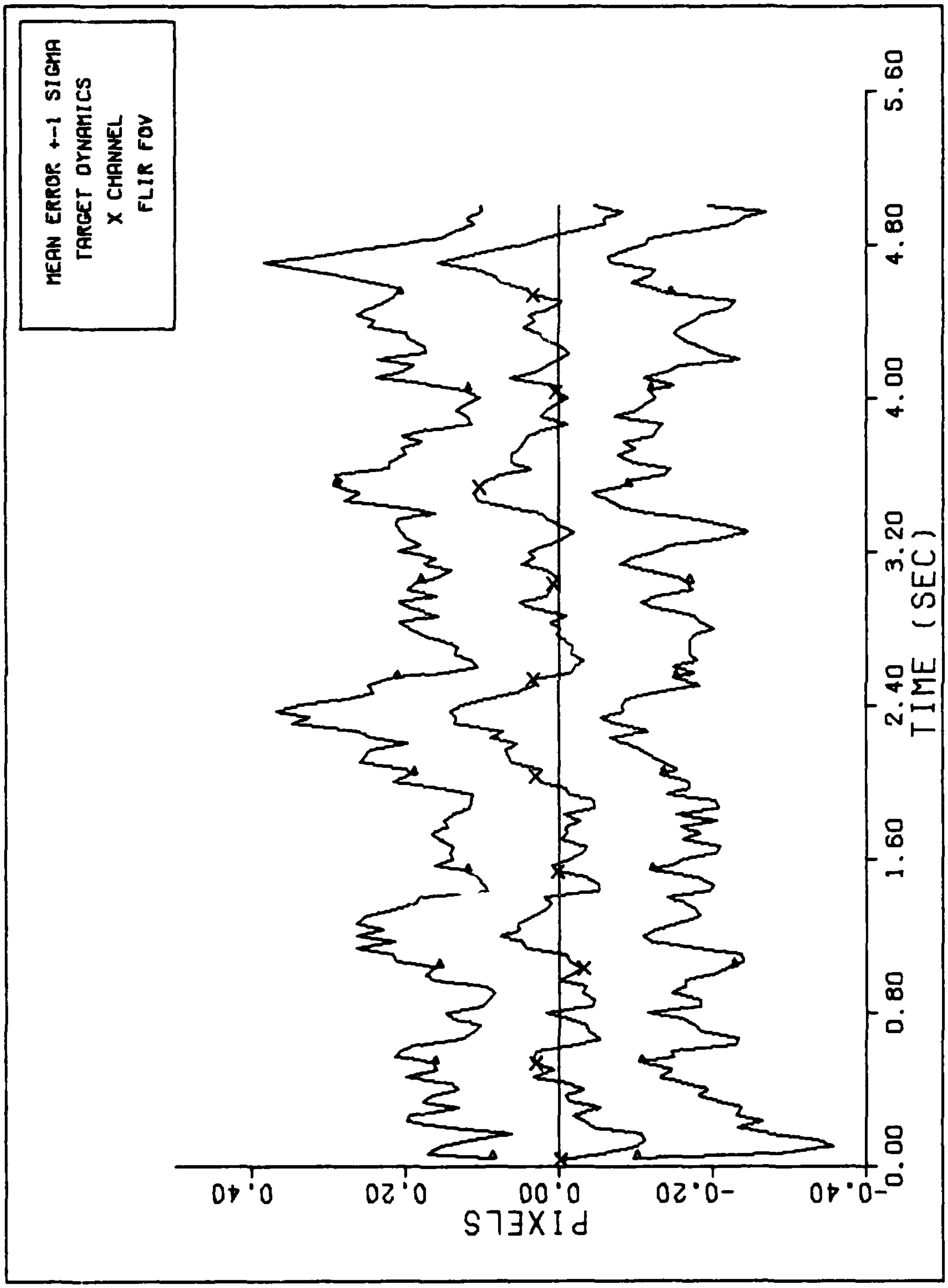
Symbol	FORTRAN Code	Valve
σ_D	SIGSI	0
$\sigma_{\bar{f}}$	SIGFLR	0
σ_A	SIGAT	0.2
$\sigma_{g_2 T}$	SIGMS	1
AR_T	ASPRO	5
	ISPTL	'NO'
	NRUN	20
	TFINAL	5
$\sigma_{\dot{x}}$	SIGMFO	-3
AR_F	ARO	1
σ_{AF}	SIGF2	0.2
σ_v	RF	2

Input Parameters

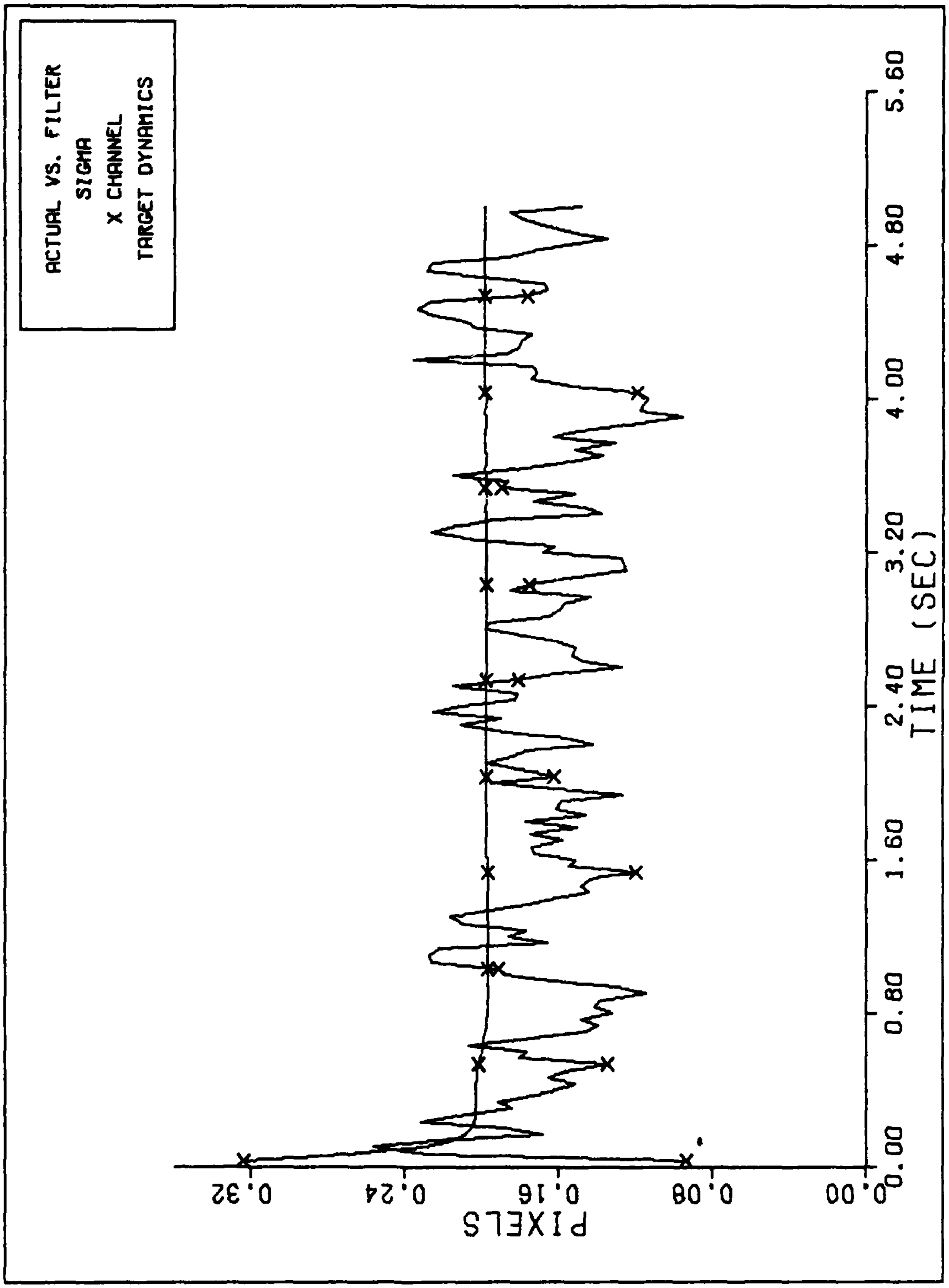
Symbol	Fortran Code / Case	E1	E2	E3	E4	E5	E6	E7
I_{max}	IMAX	25	25	25	25	25	25	25
σ_N	SIGMAB	2	2	2	2	2	2	2
$I_{max}F$	FIMAXO	25	25	25	25	25	25	25
σ_{DF}	SIGFIO	500	1000	1000	5000	1000	2000	10,000
Acquisition QFD		NO	NO	NO	NO	NO	NO	YES
Turn SIGF1		-	-	-	-	20,000	50,000	-
S/N	SN	12.5	12.5	12.5	12.5	12.5	12.5	12.5
Turn (G)		0	5	10	10	20	20	0
Initial Conditions $x_d(0)$	XFP(1)	0	0	0	0	0	0	1.46
$y_d(0)$	SFP(2)	0	0	0	0	0	0	1.43
$\dot{x}_e$	XDOT	-530.33	-750	-750	-750	-750	-750	-530.33
$\dot{y}_e$	YDOT	-530.33	0	0	0	0	0	-530.33
$\dot{z}_e$	ZDOT	0	0	0	0	0	0	-530.33

Input Parameters

Symbol	Case Fortran Code	E8	E9	E10	E11	E12	E13
I_{max}	IMAX	25	25	10	4	25	10
σ_N	SIGMAB	2	2	2	2	12.5	5
$I_{max}F$	FIMAX0	25	25	10	4	25	20
σ_{DF}	SIGF10	10,000	10,000	500	500	20,000	20,000
Acquisition Q_{FD}		YES	YES	NO	NO	NO	NO
Turn SIGFI		-	-	-	-	-	-
S/N	SN	12.5	12.5	5	2	2	2
Turn (G)		0	0	0	0	0	0
Initial Conditions $x_d(0)$	XFP(1)	0	0.5	0	0	0	0
$y_d(0)$	SFP(2)	0	0.5	0	0	0	0
$\dot{x}_e$	XDOT	-517.55	-525	-530.33	-530.33	-530.33	-530.33
$\dot{y}_e$	YDOT	-543.11	-535	-530.33	-530.33	-530.33	-530.33
$\dot{z}_e$	ZDOT	5	8	0	0	0	0

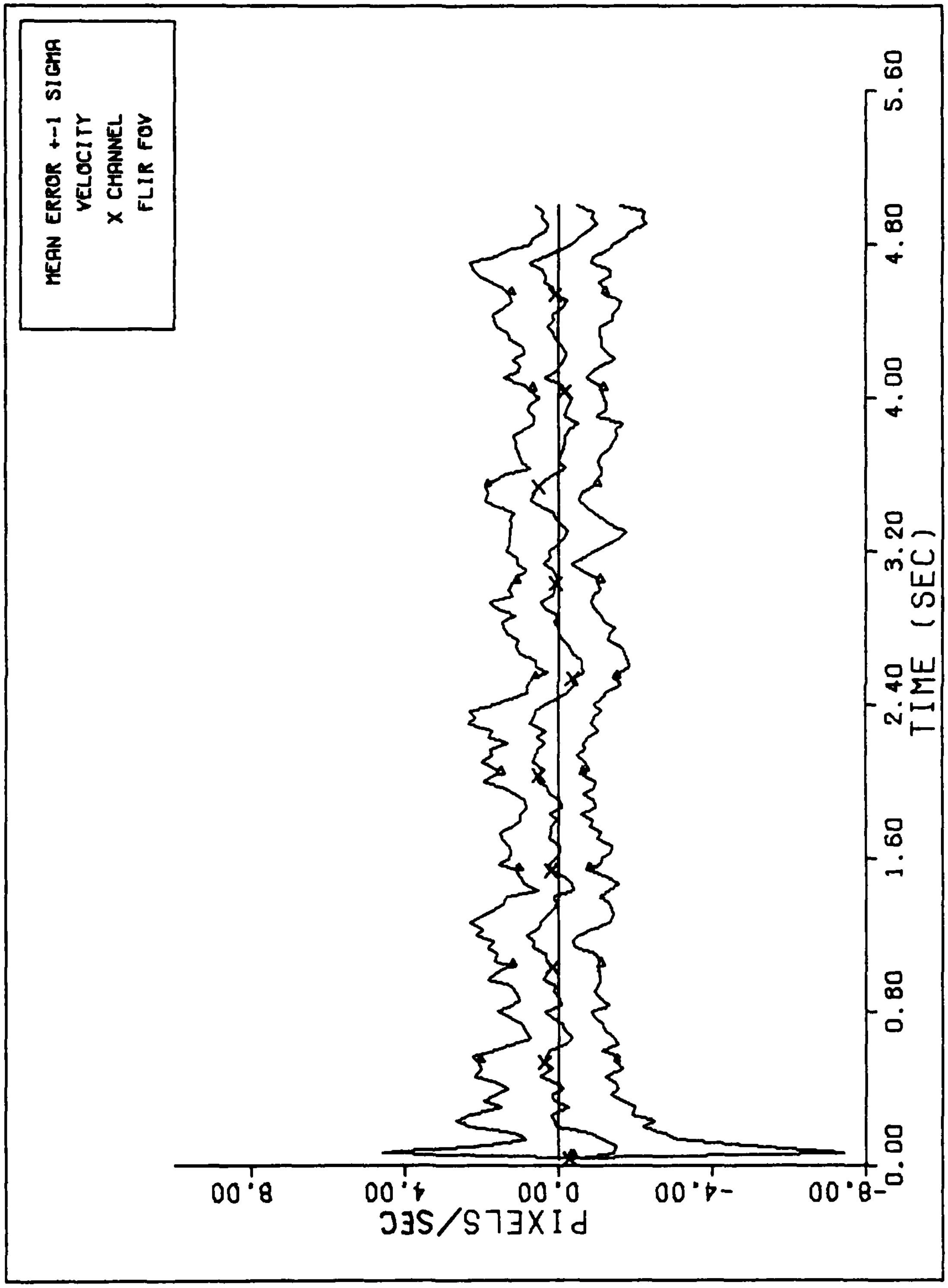


X CHANNEL DYNAMICS ERROR (S/N=12.5)
Figure E-1a



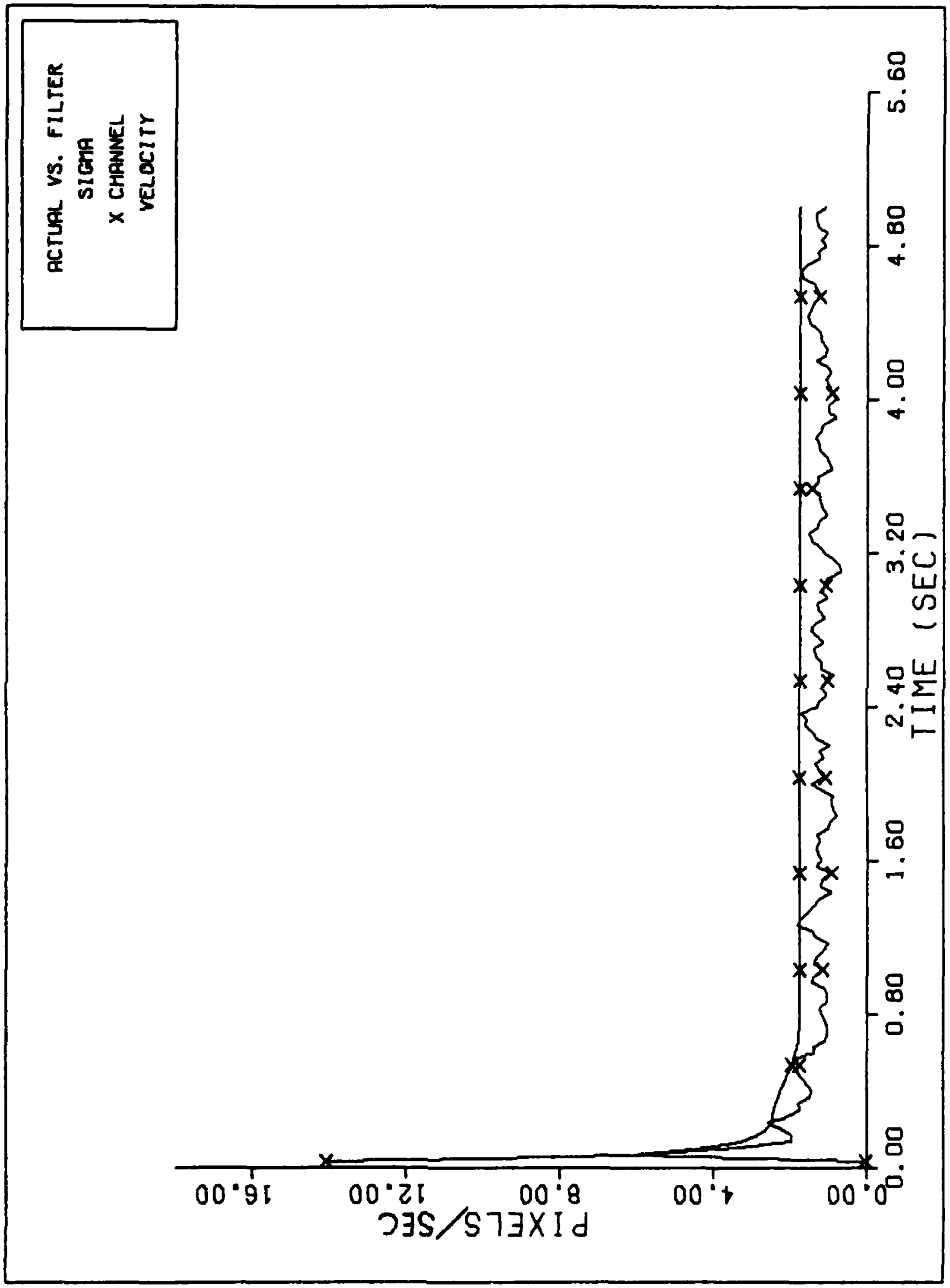
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-1b



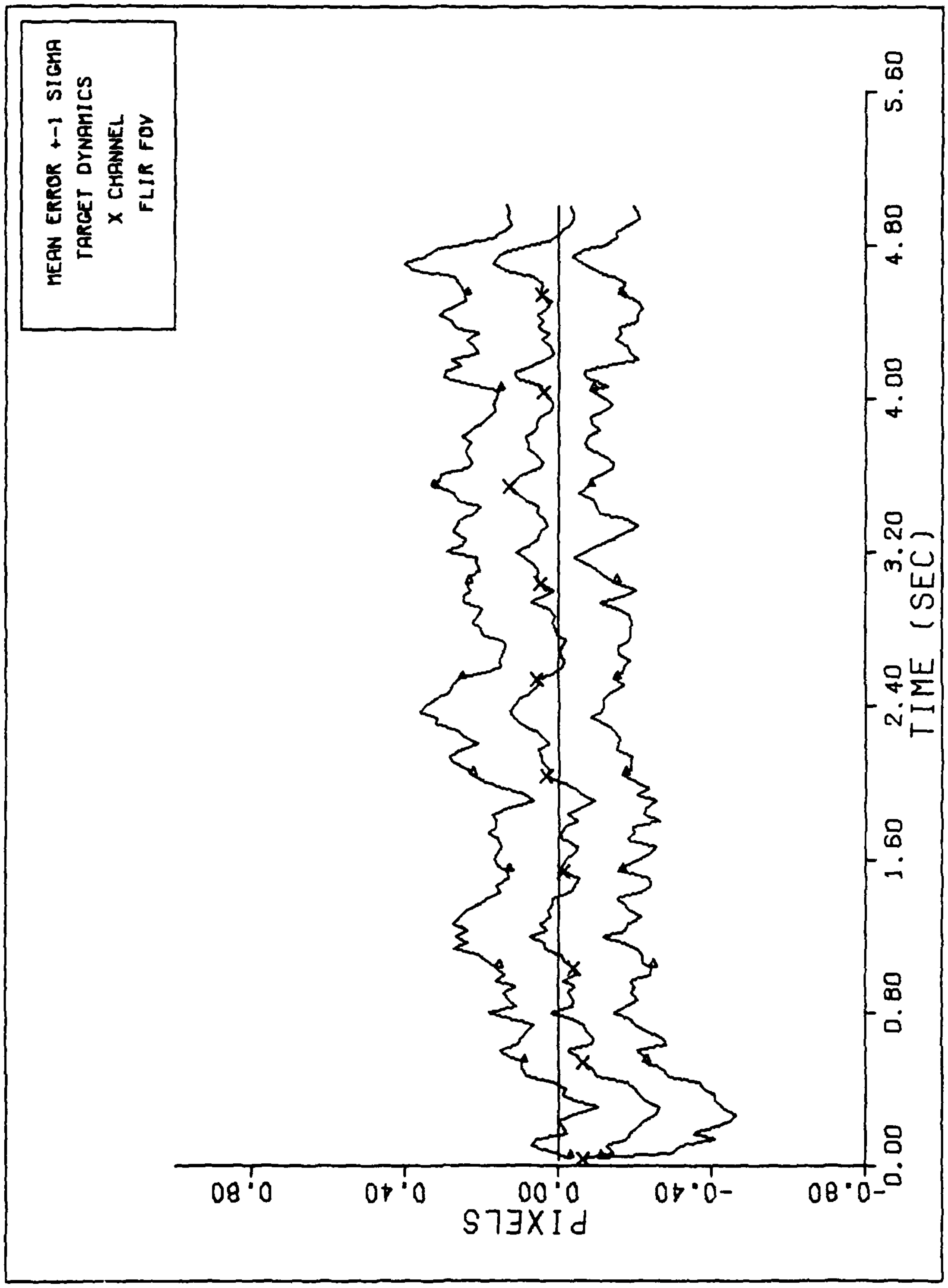
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure E-1c



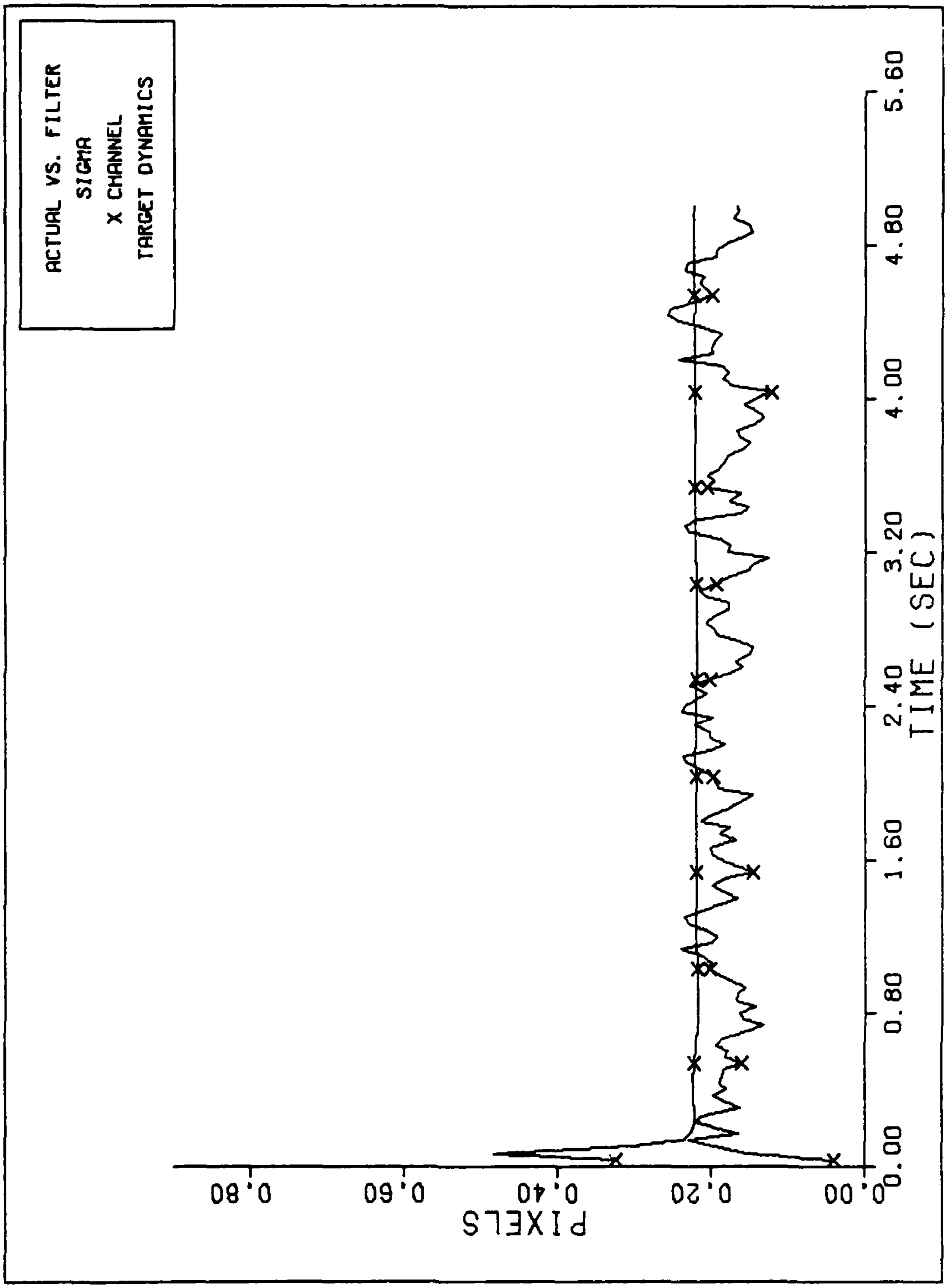
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-1d



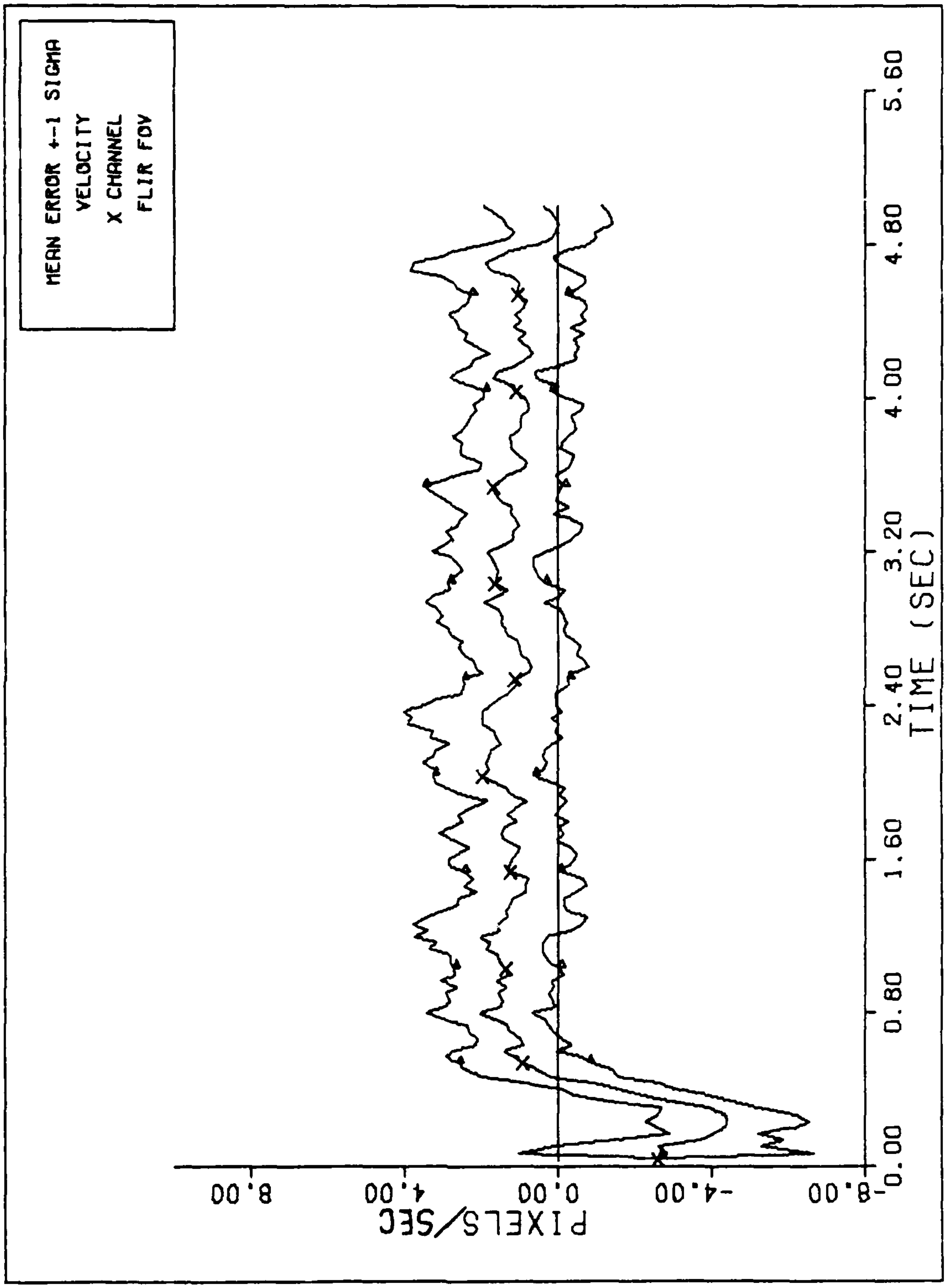
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-2a



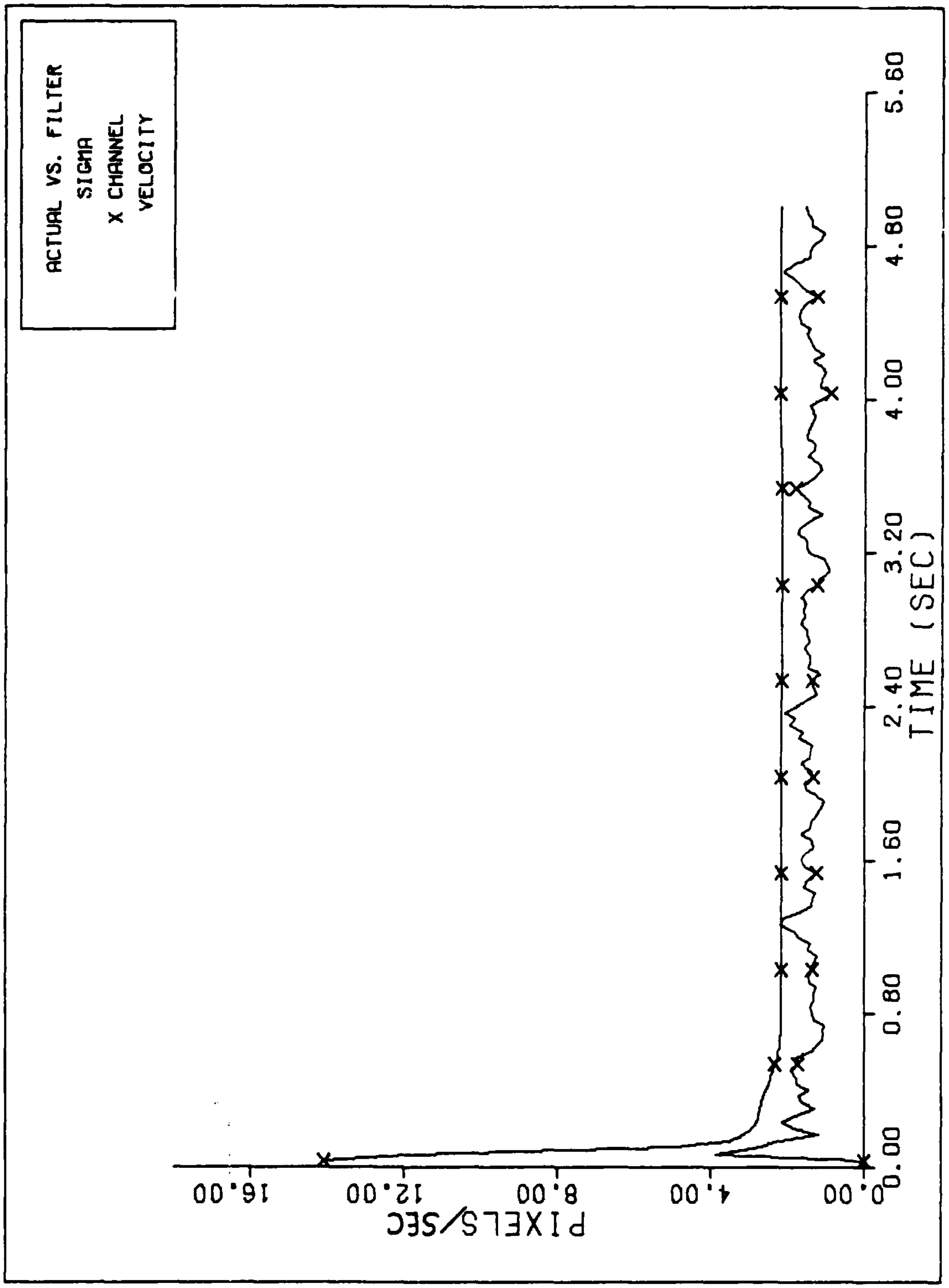
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-2b



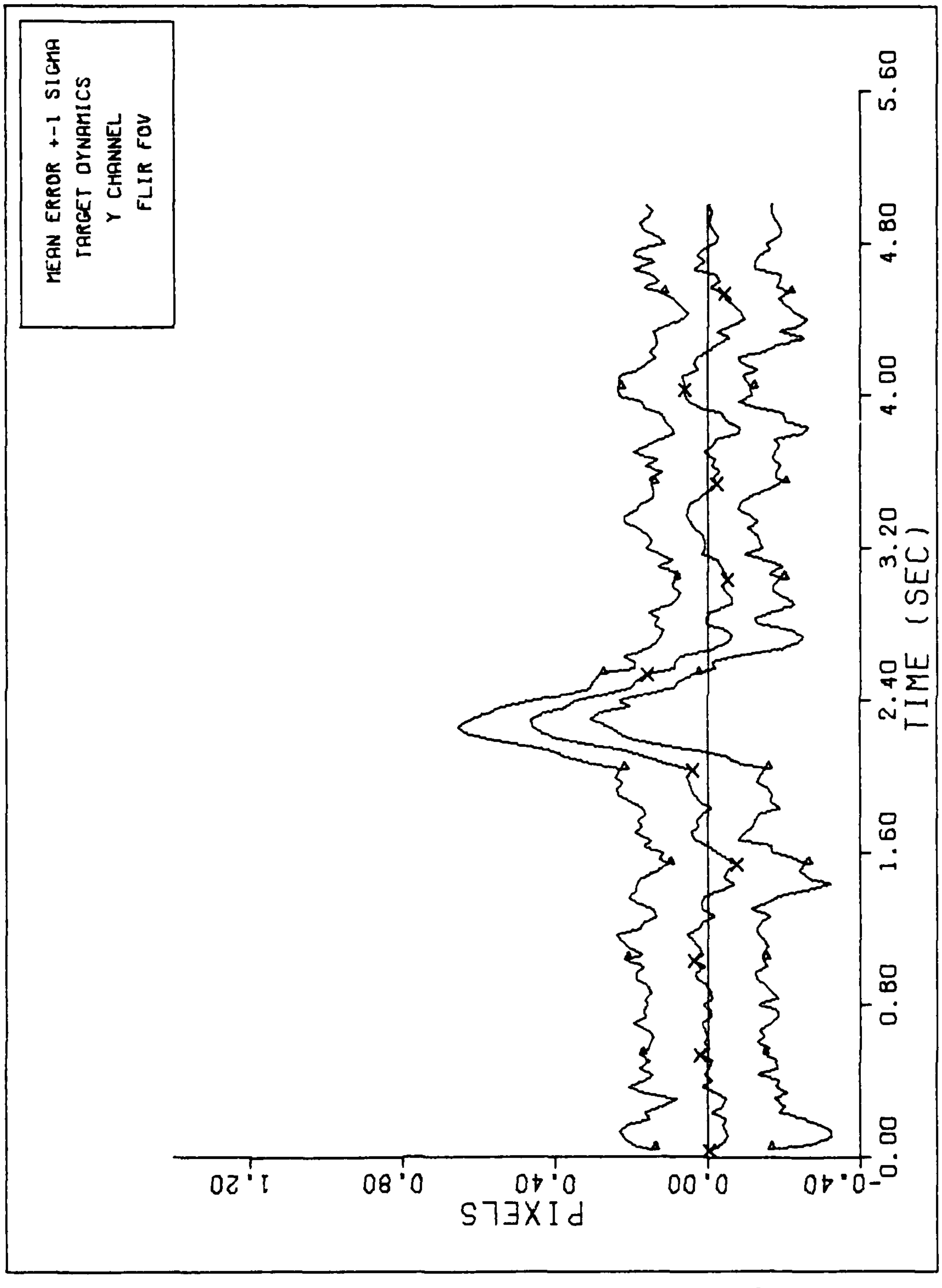
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure E-2c



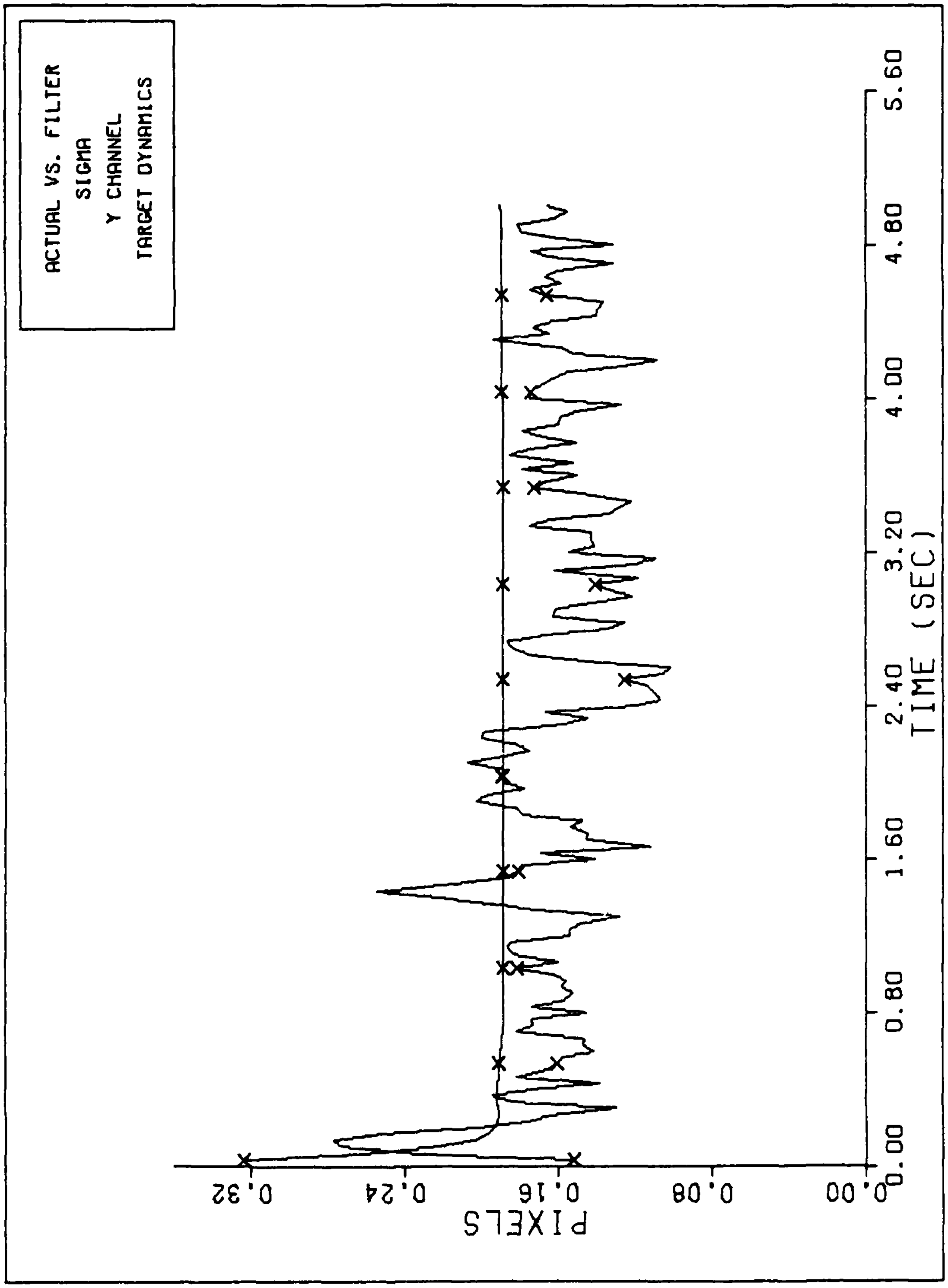
FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure E-2d



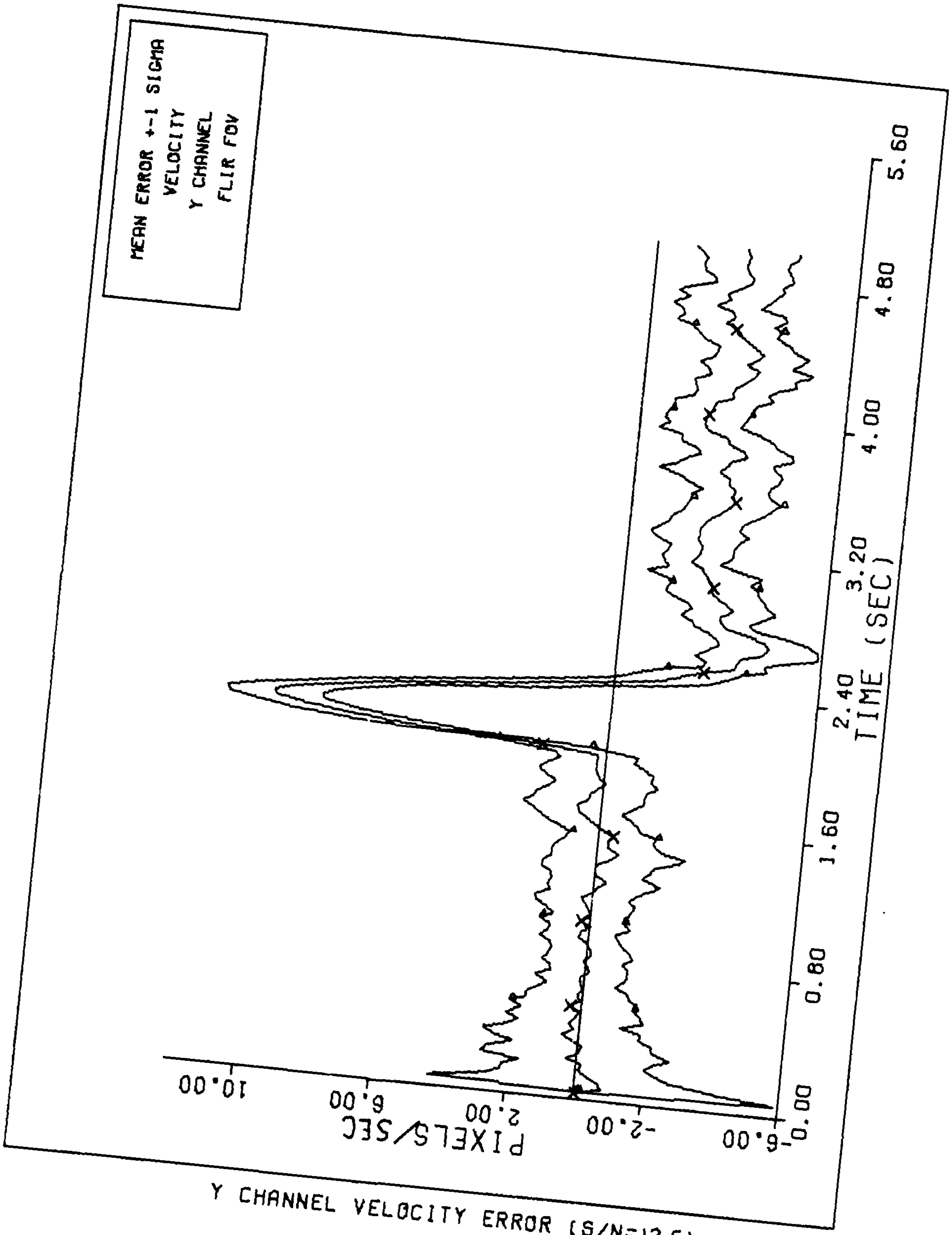
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-2e



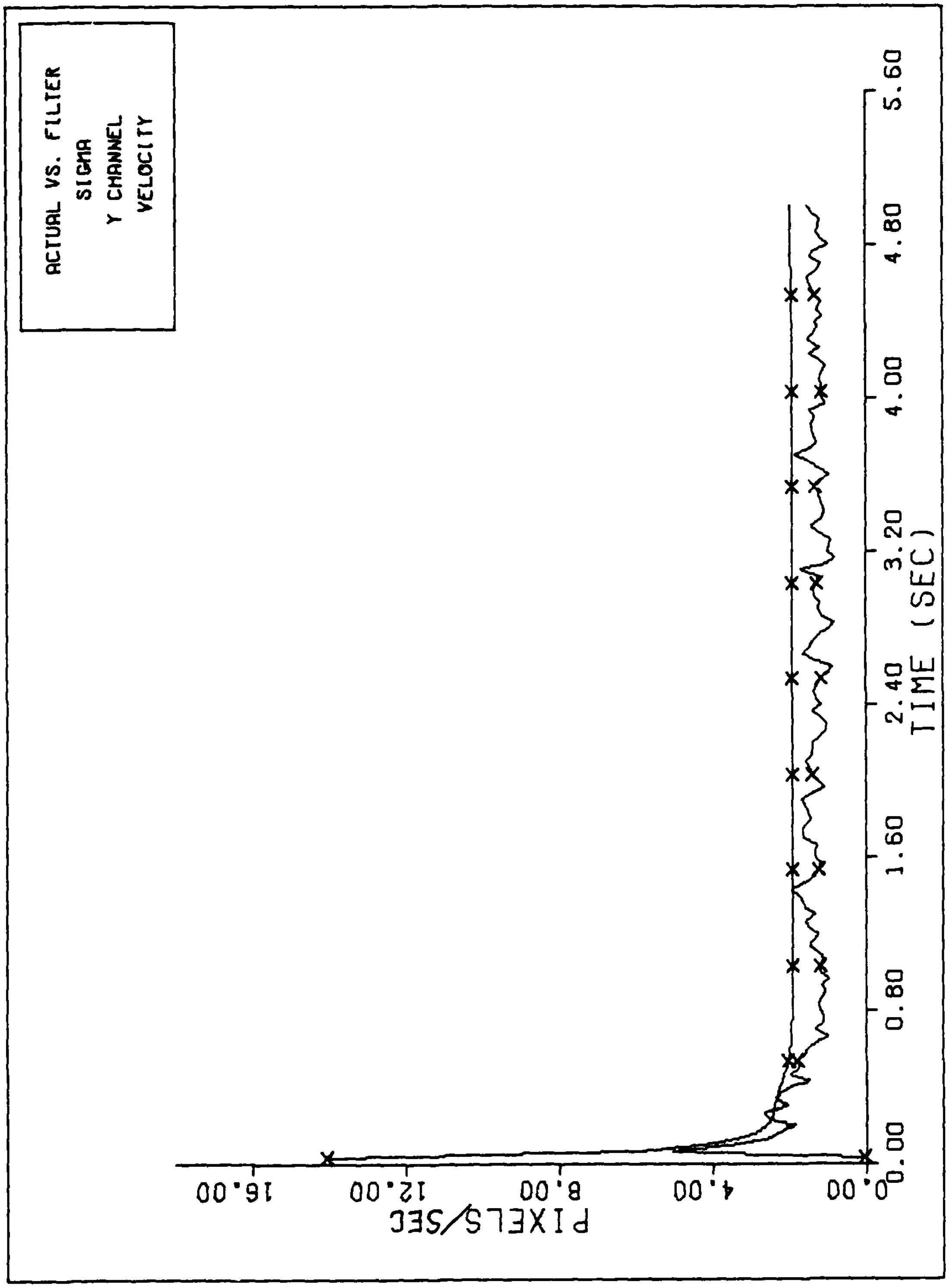
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-2f



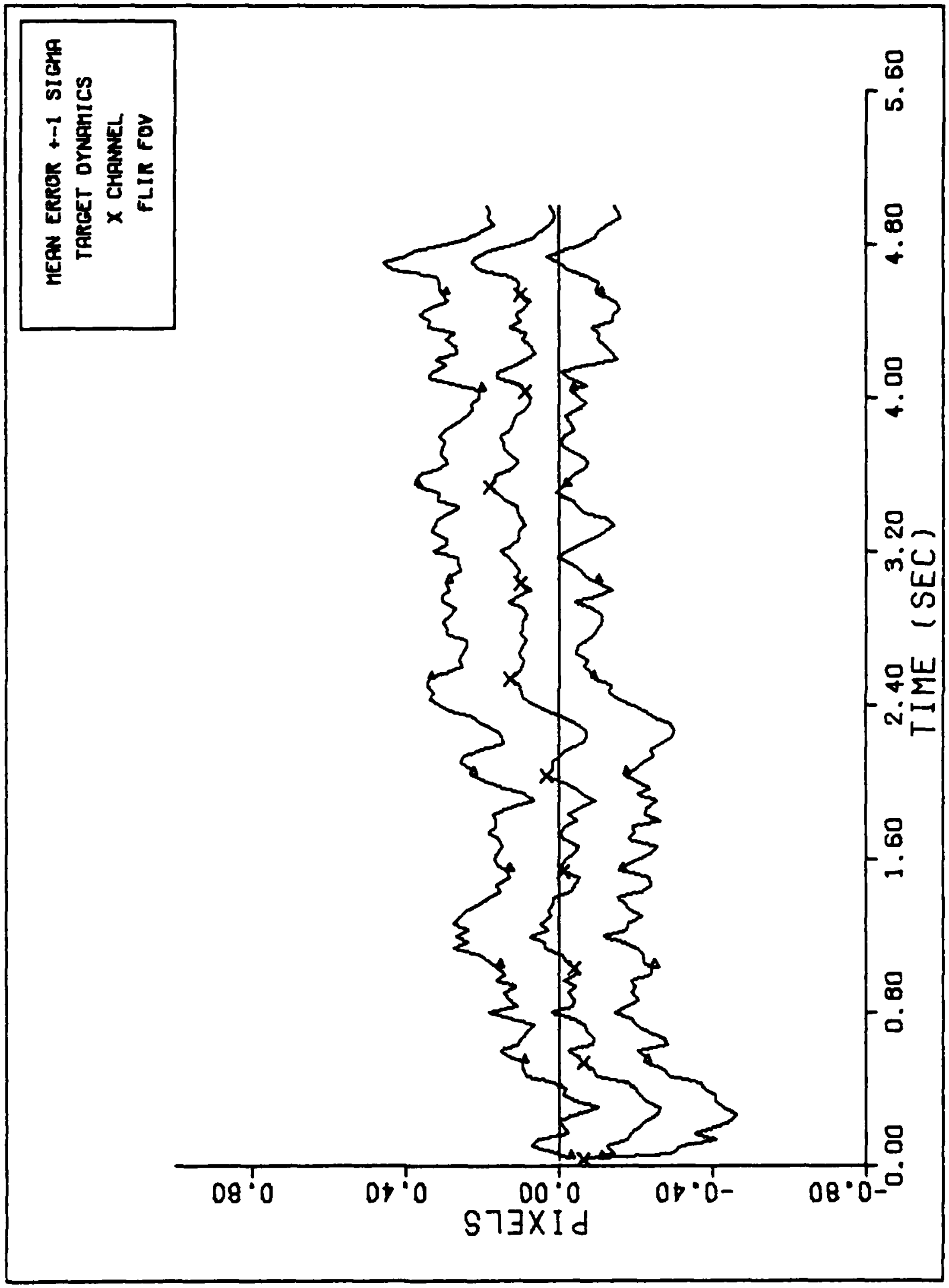
Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure E-2g



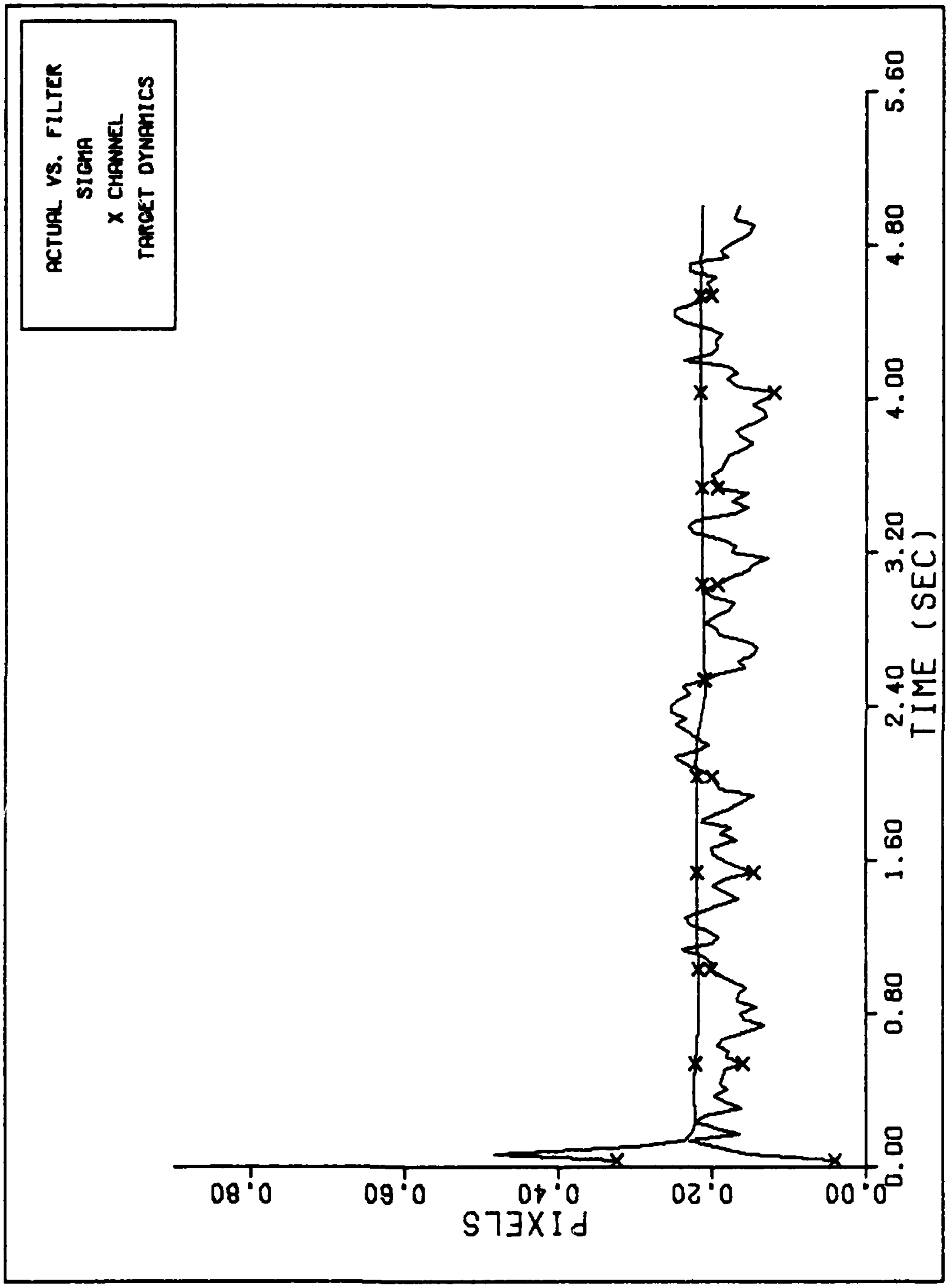
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-2h



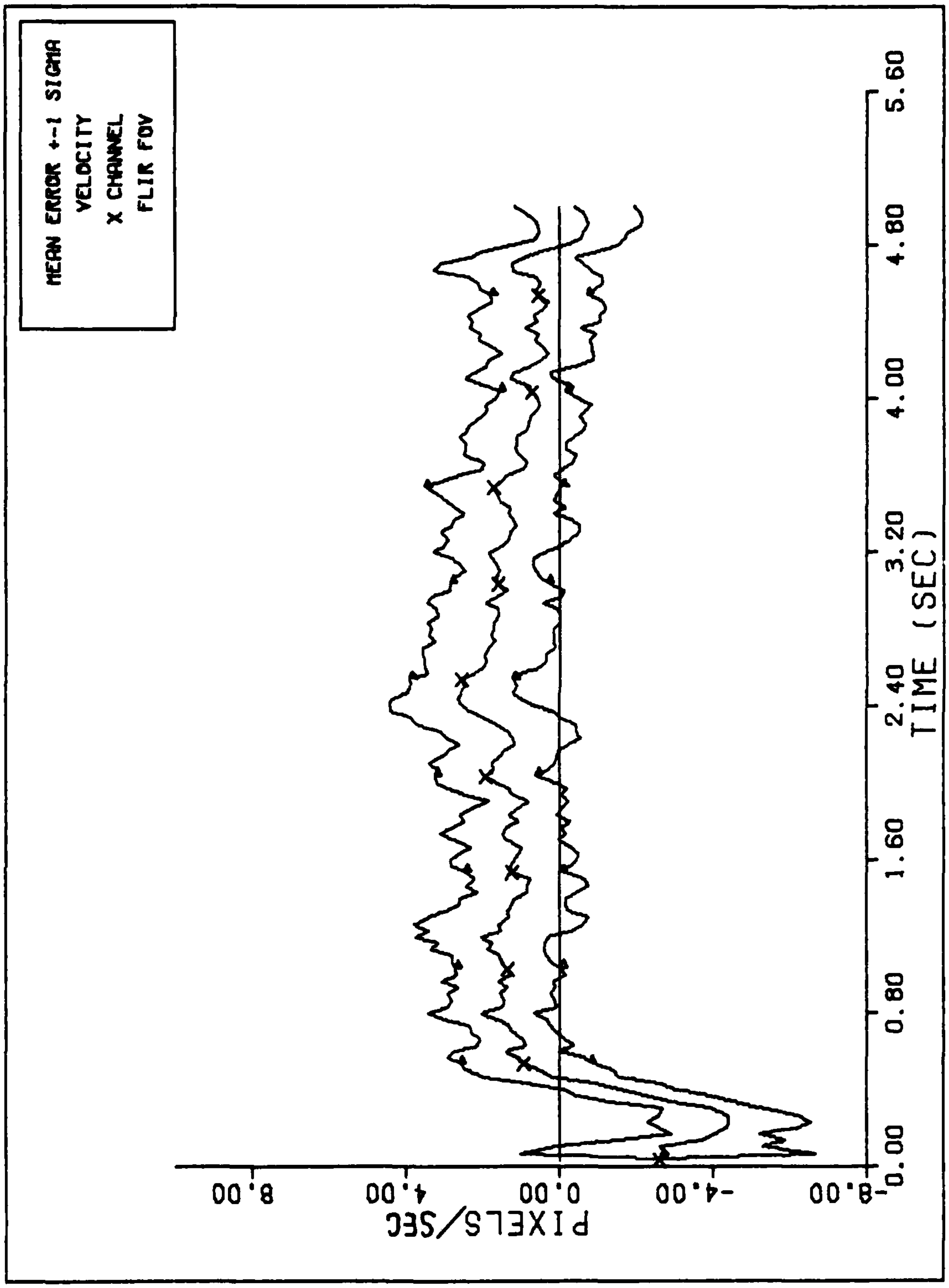
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-3a



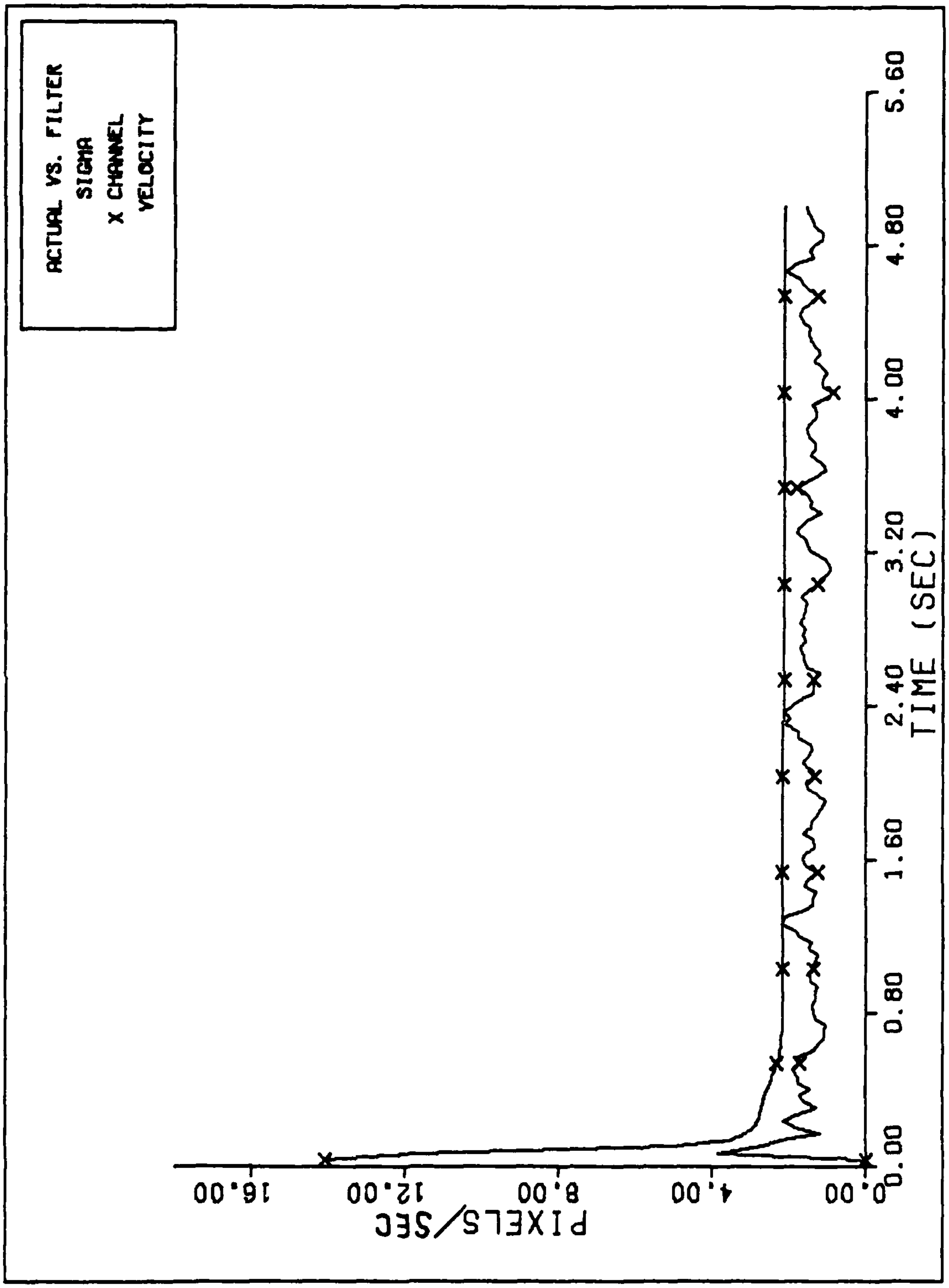
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-3b



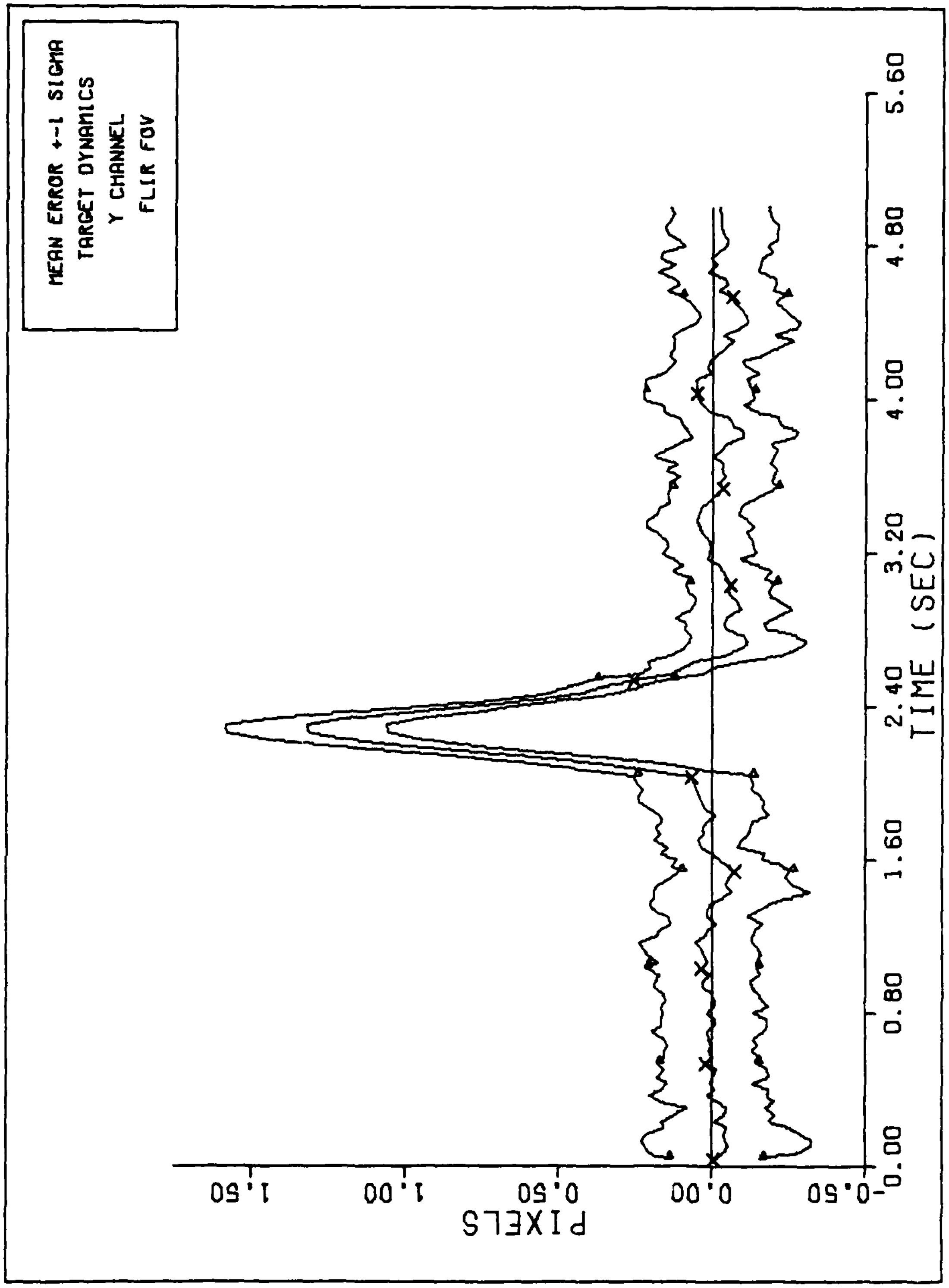
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure E-3c



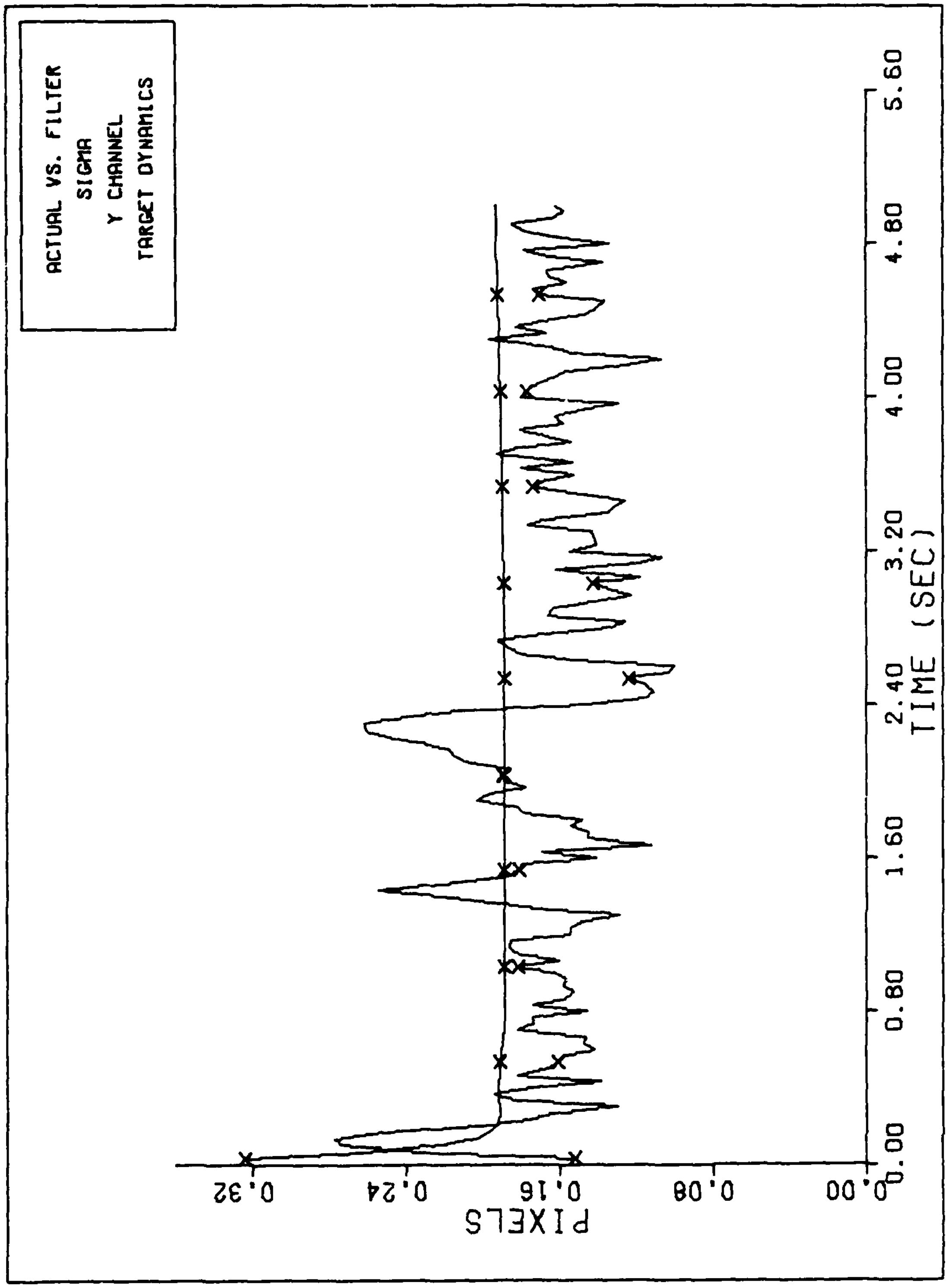
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-3d



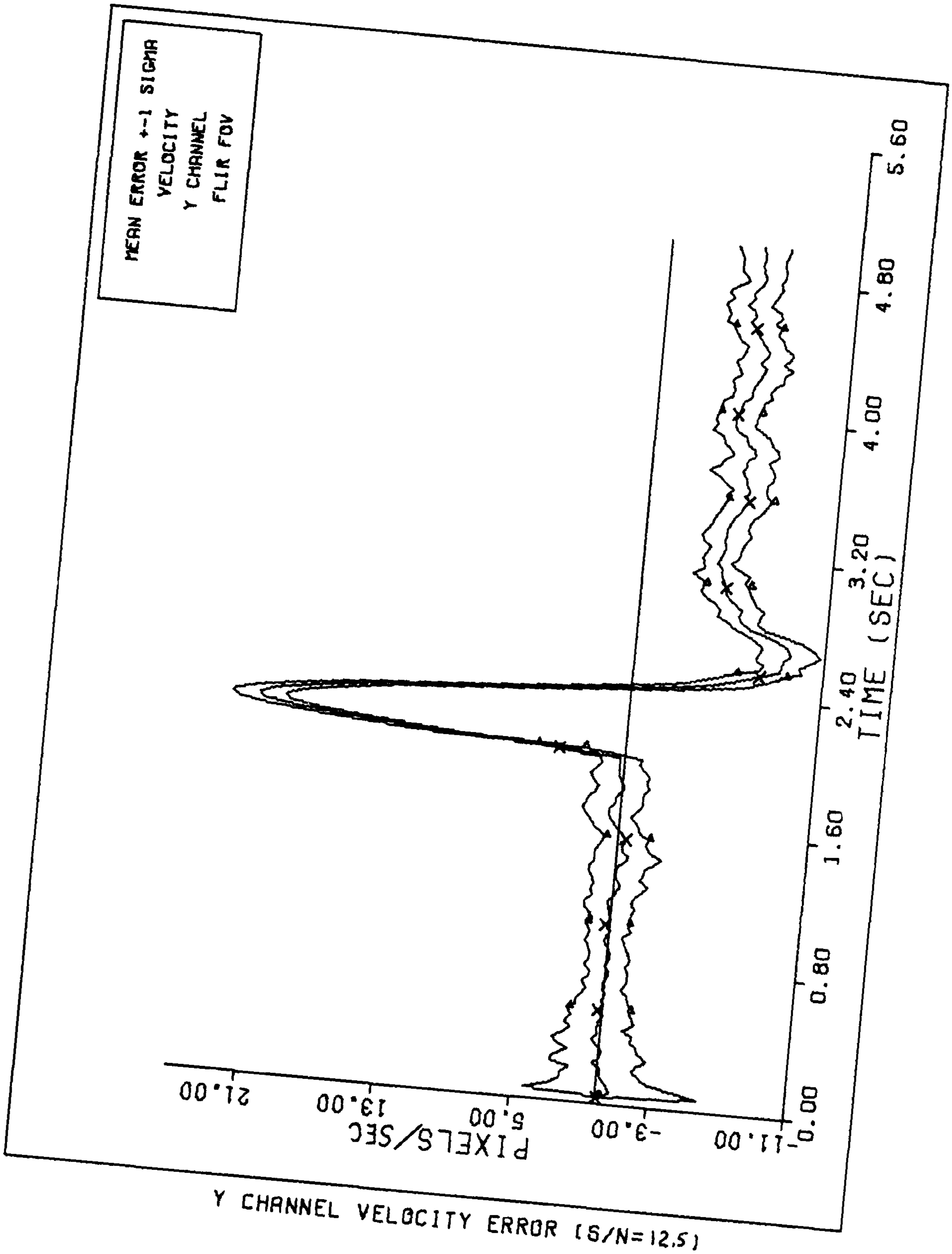
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-3e



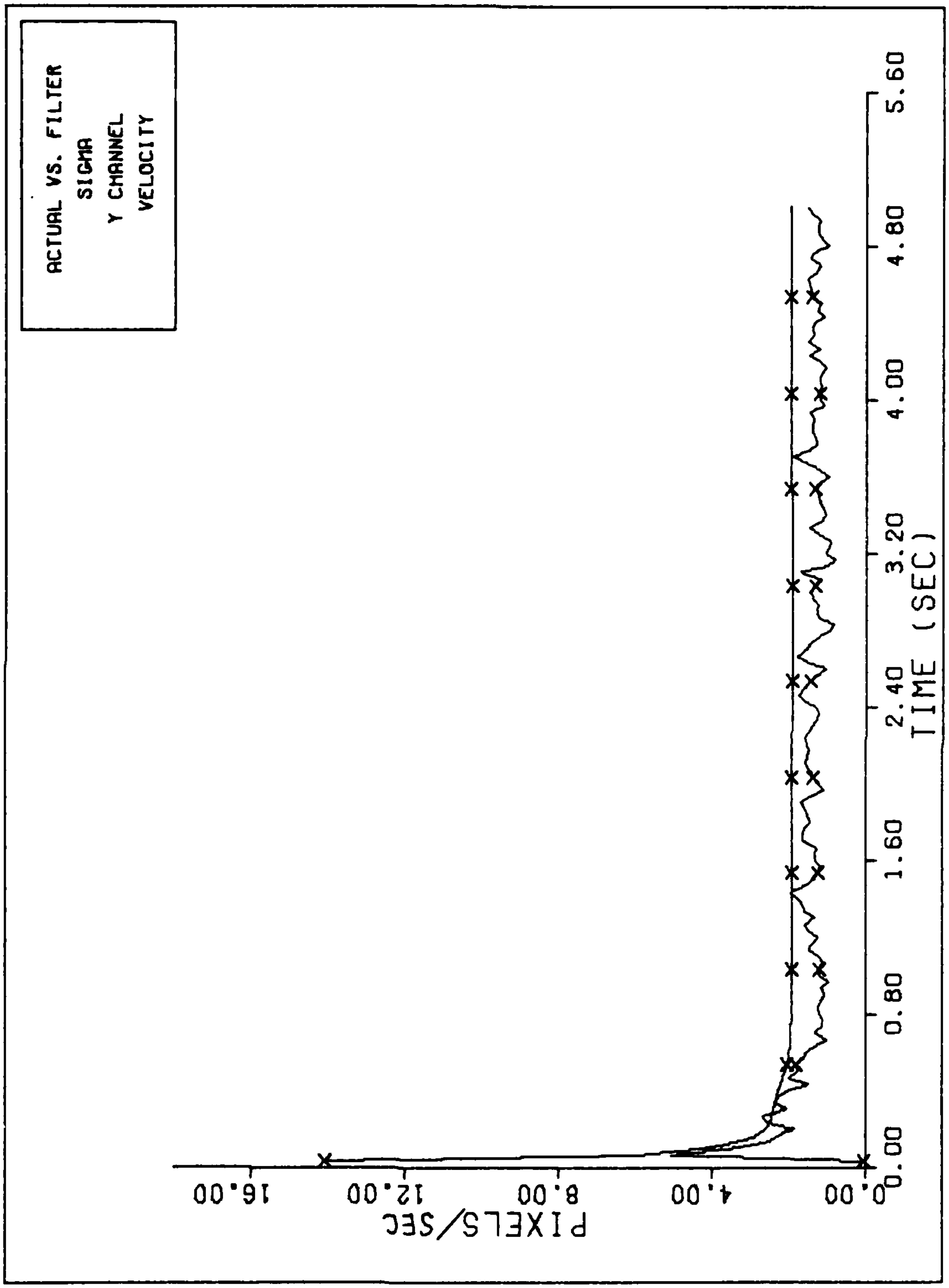
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-3f



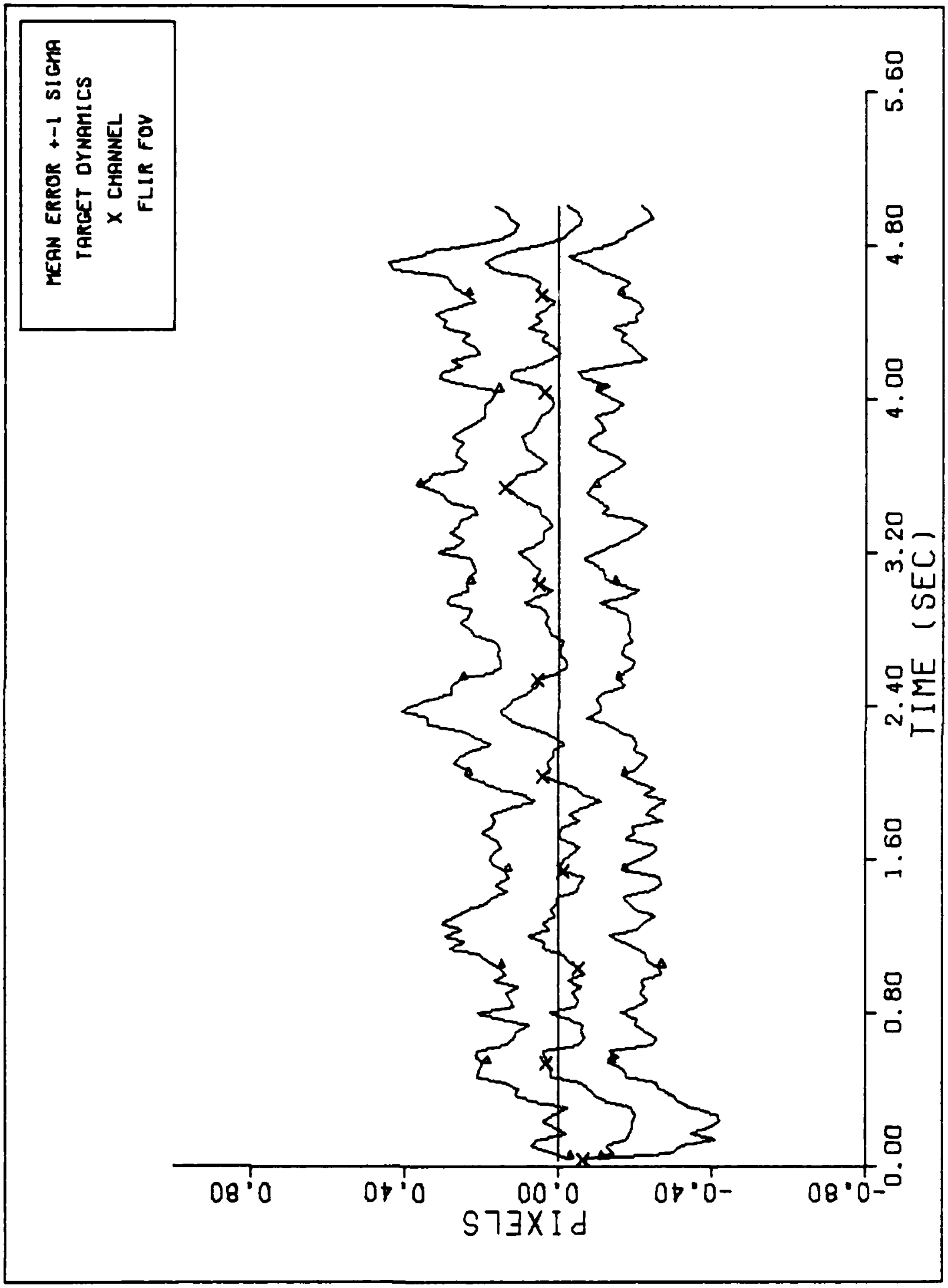
Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure E-3g



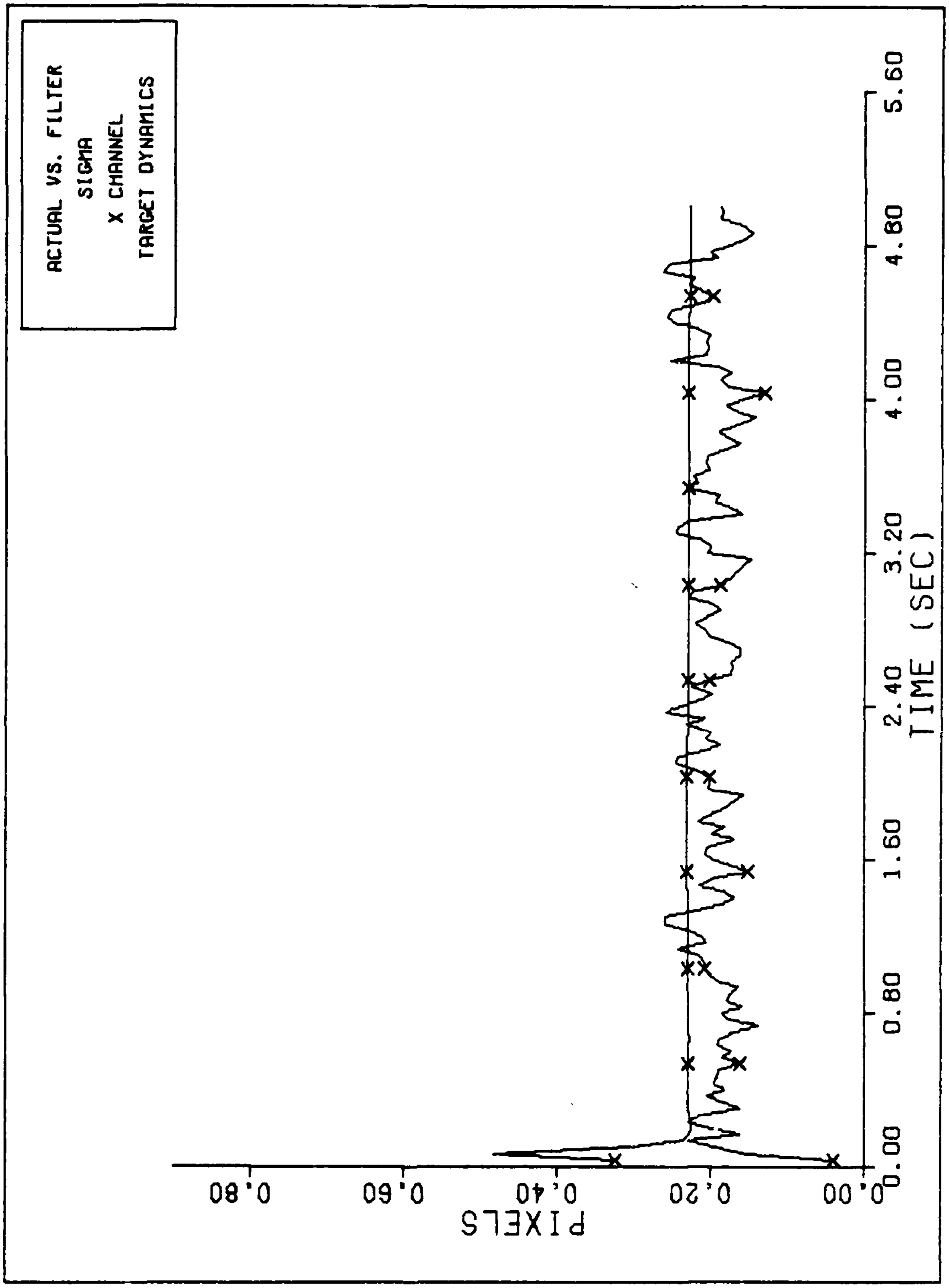
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-3h



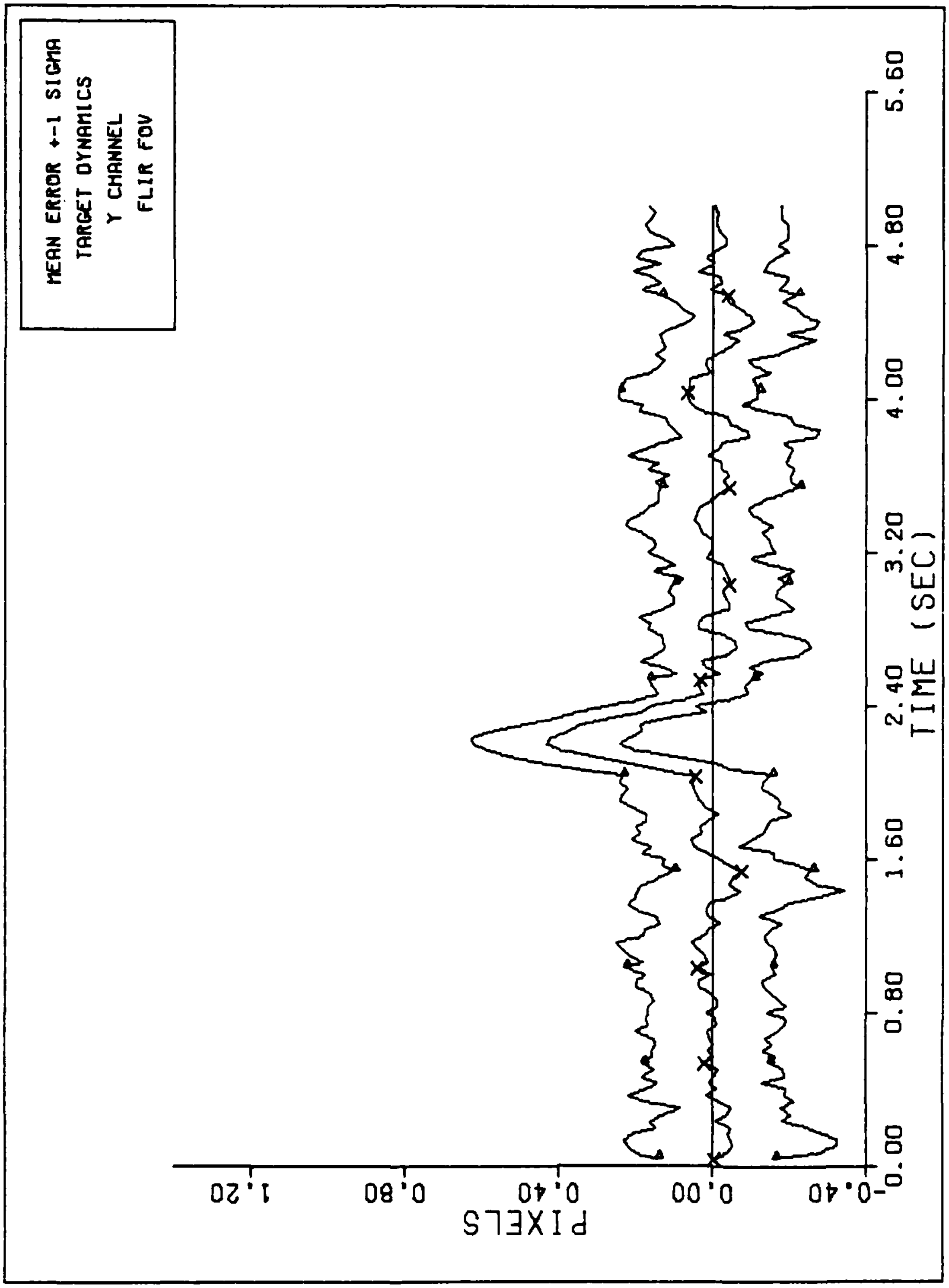
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-4a



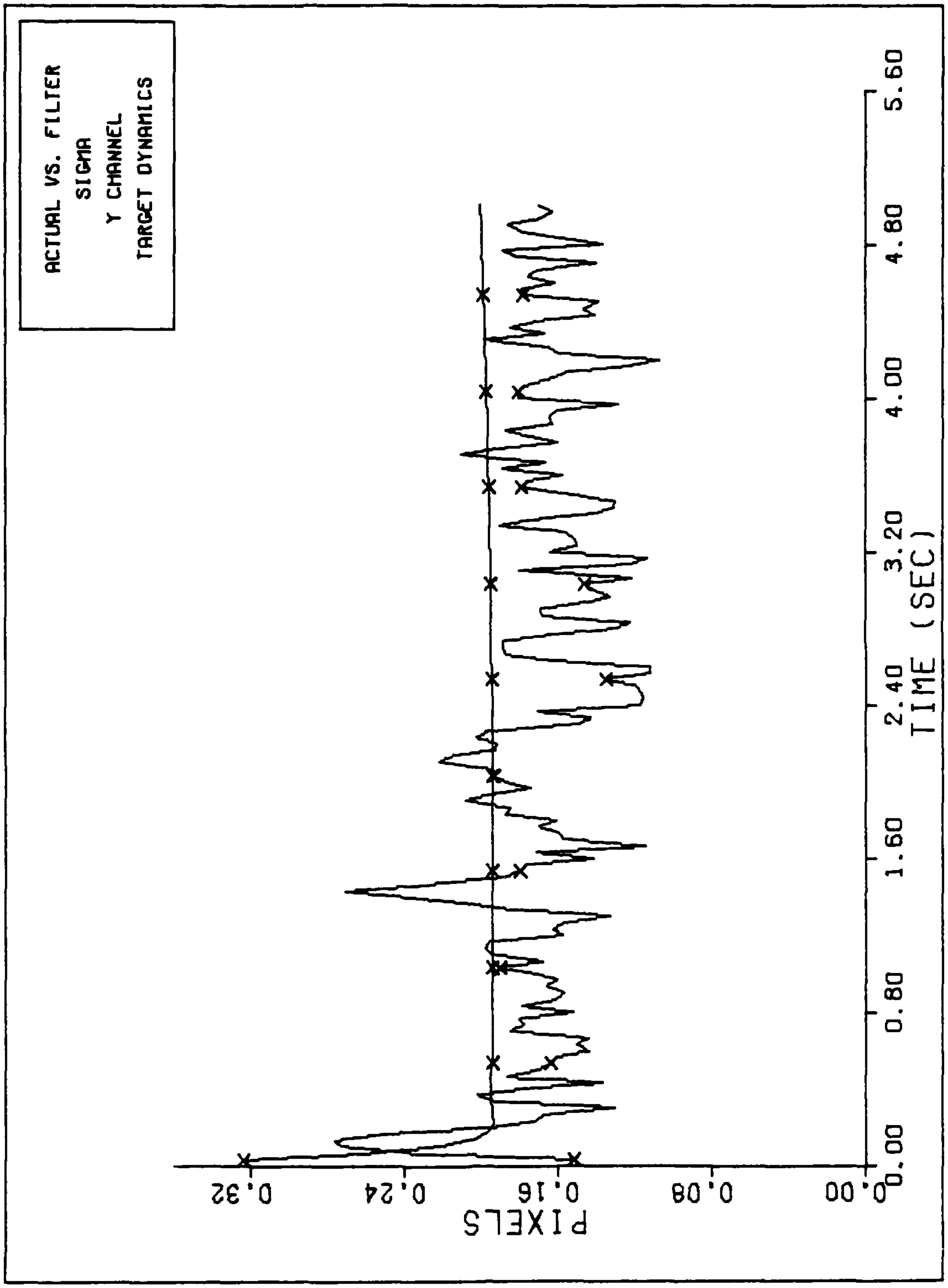
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-4b



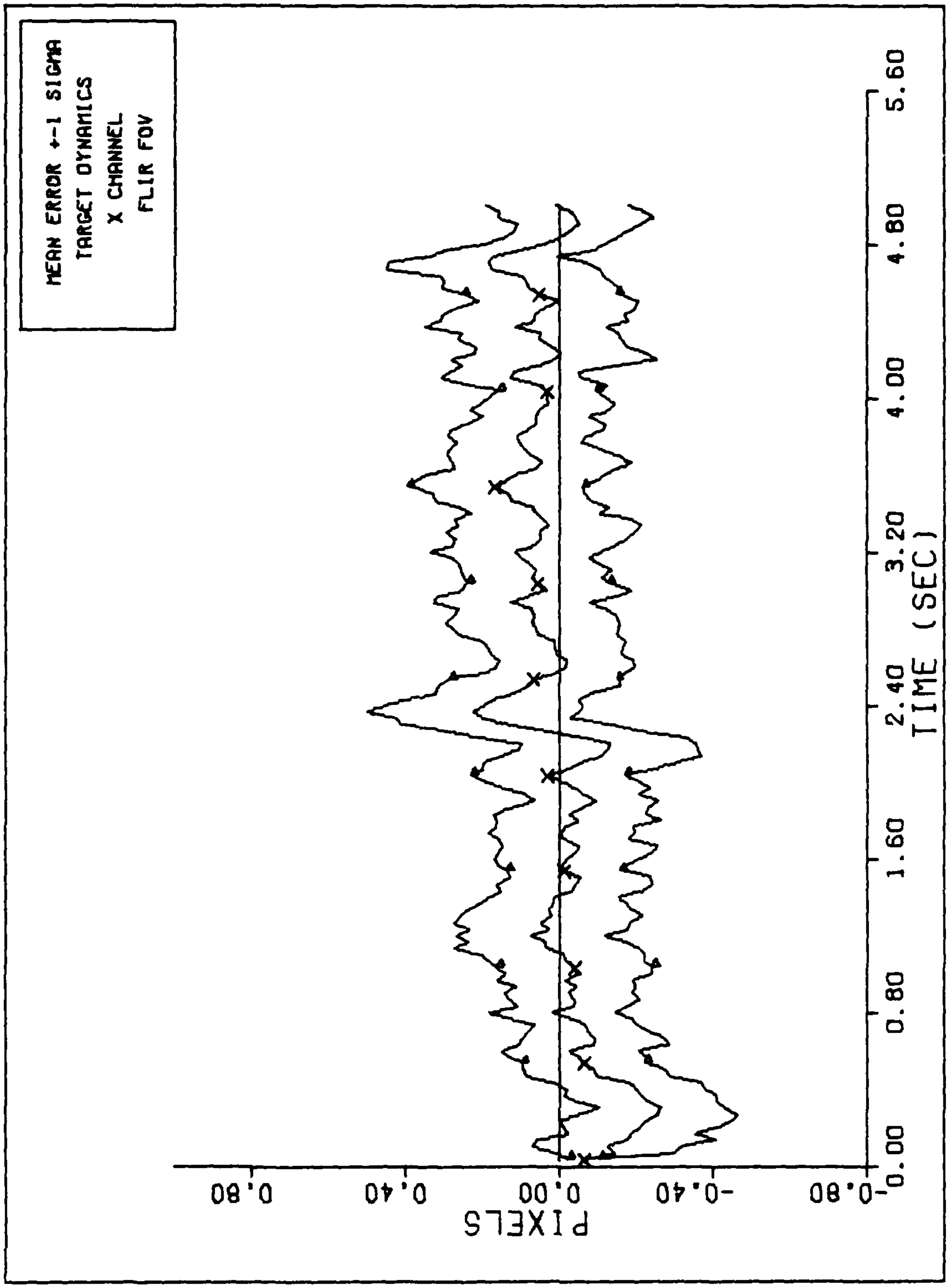
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-4c



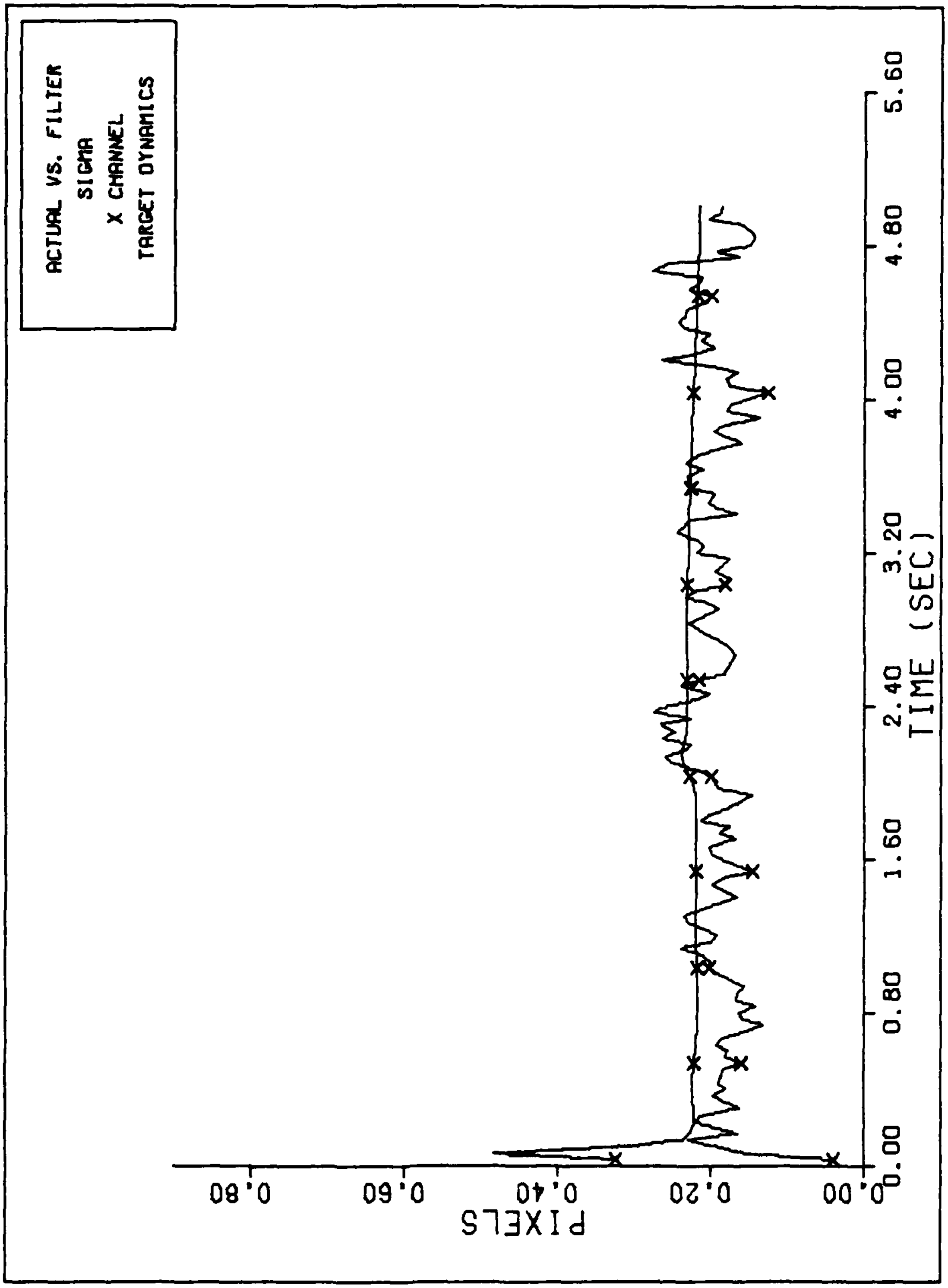
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-4d



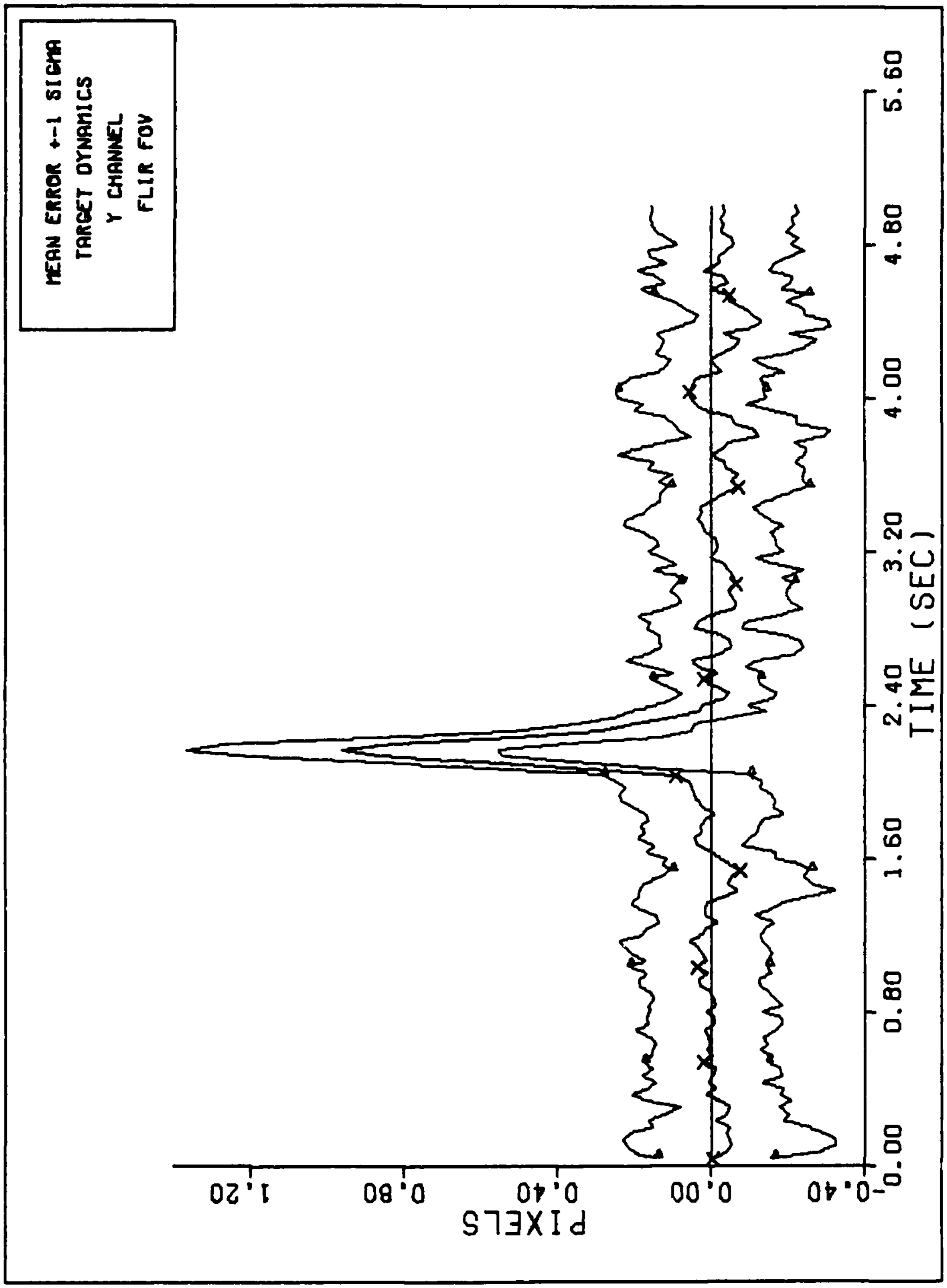
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-5a



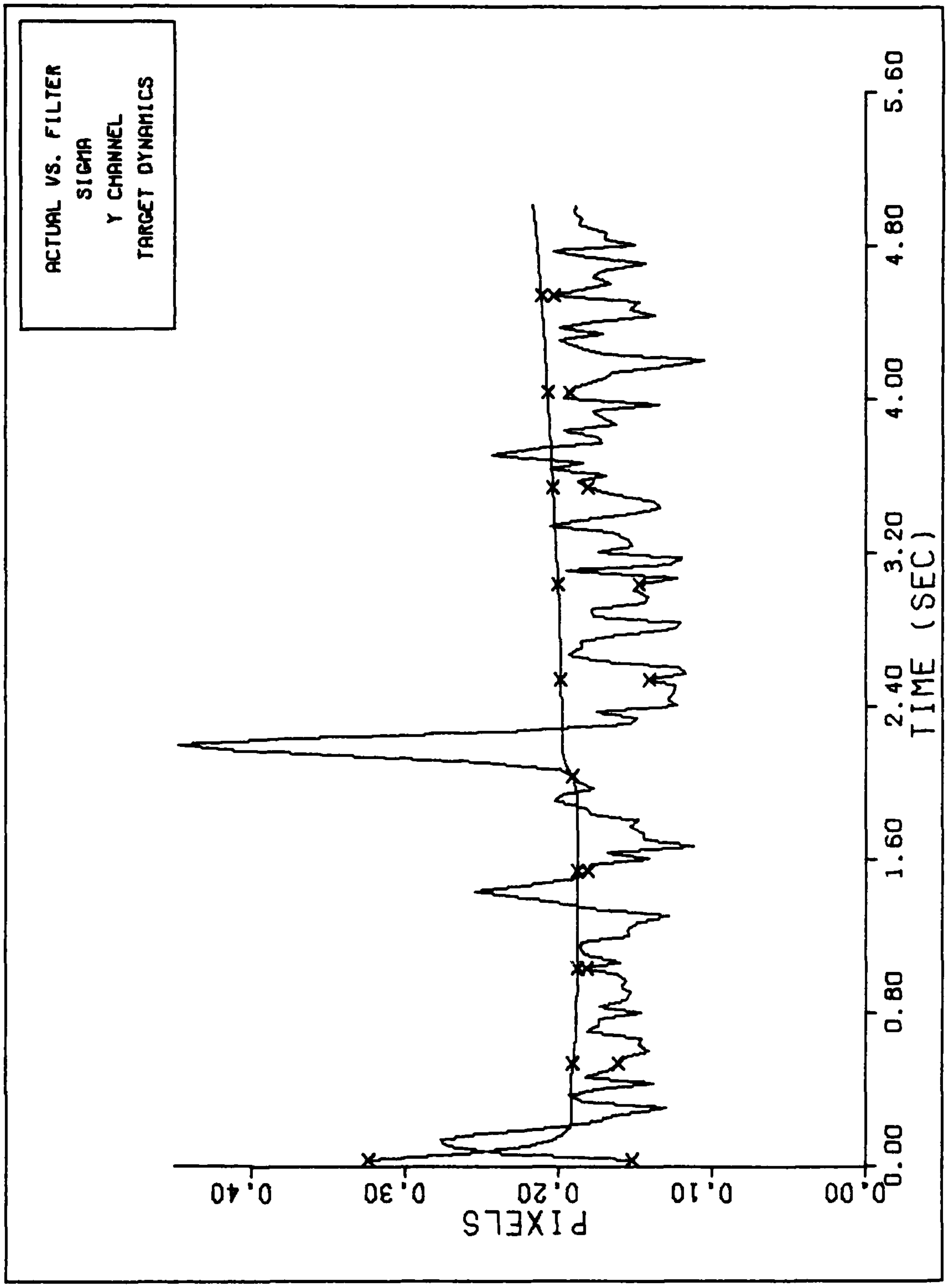
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-5b



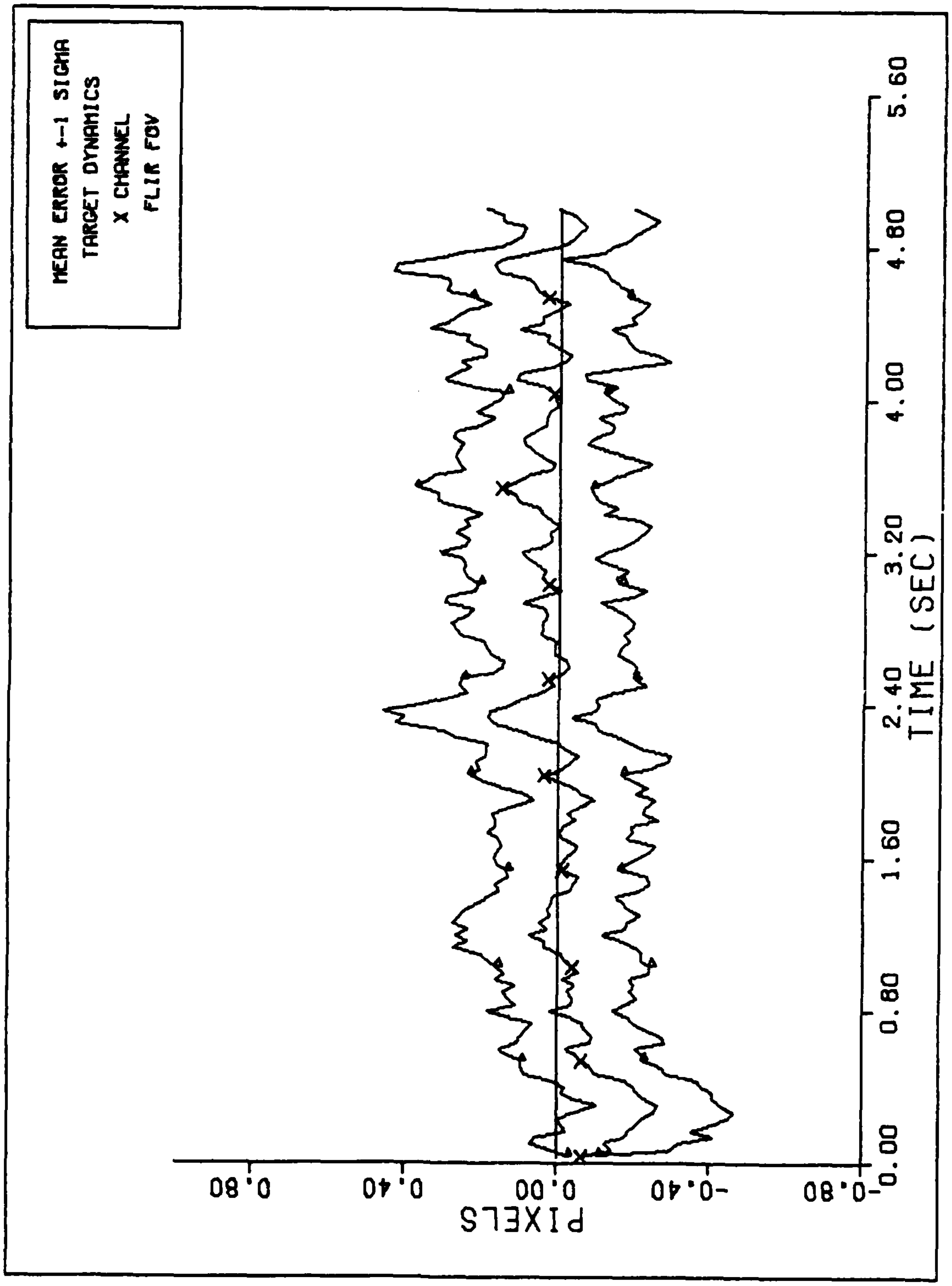
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-5c



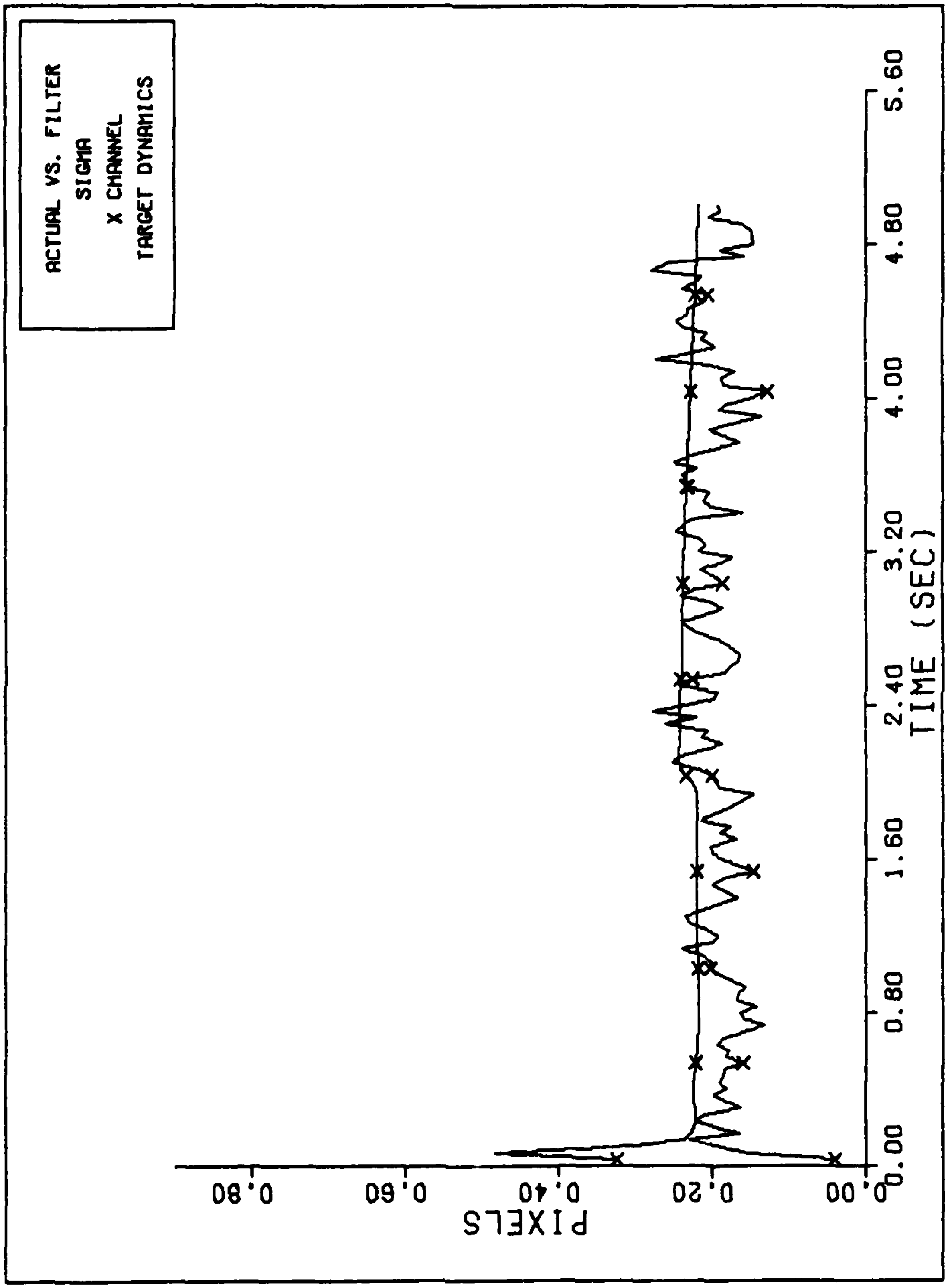
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-5d



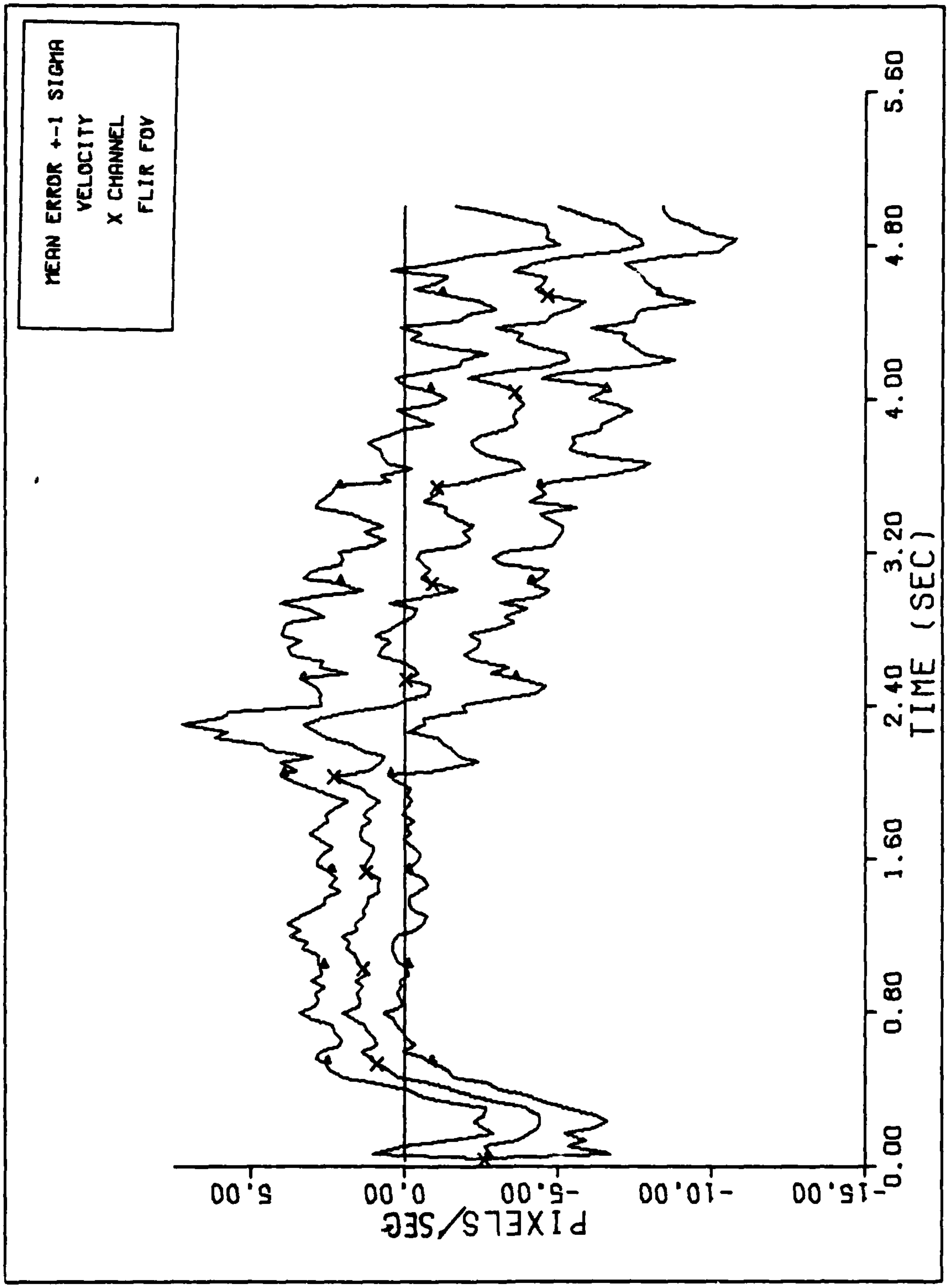
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-6a



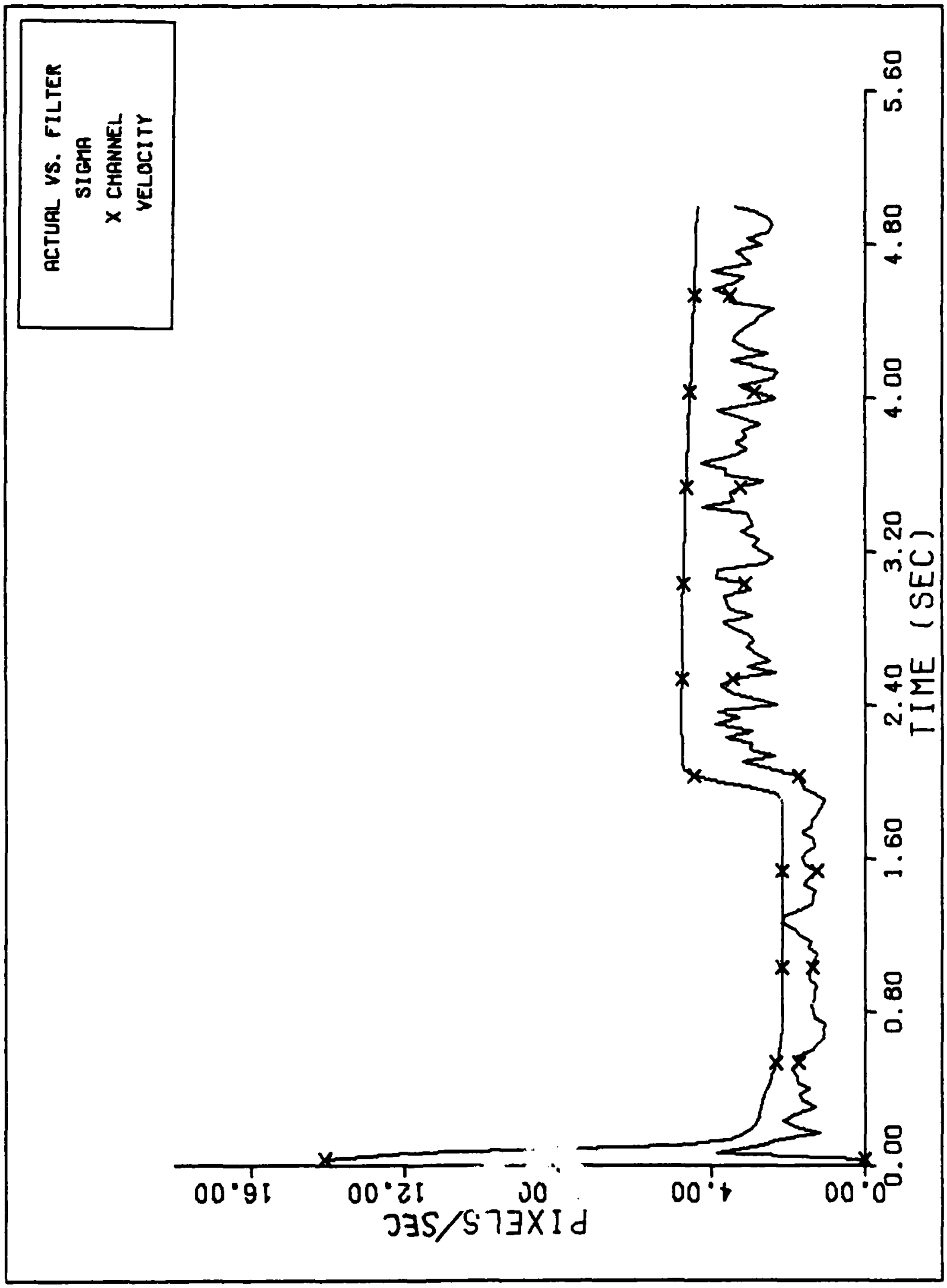
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-6b



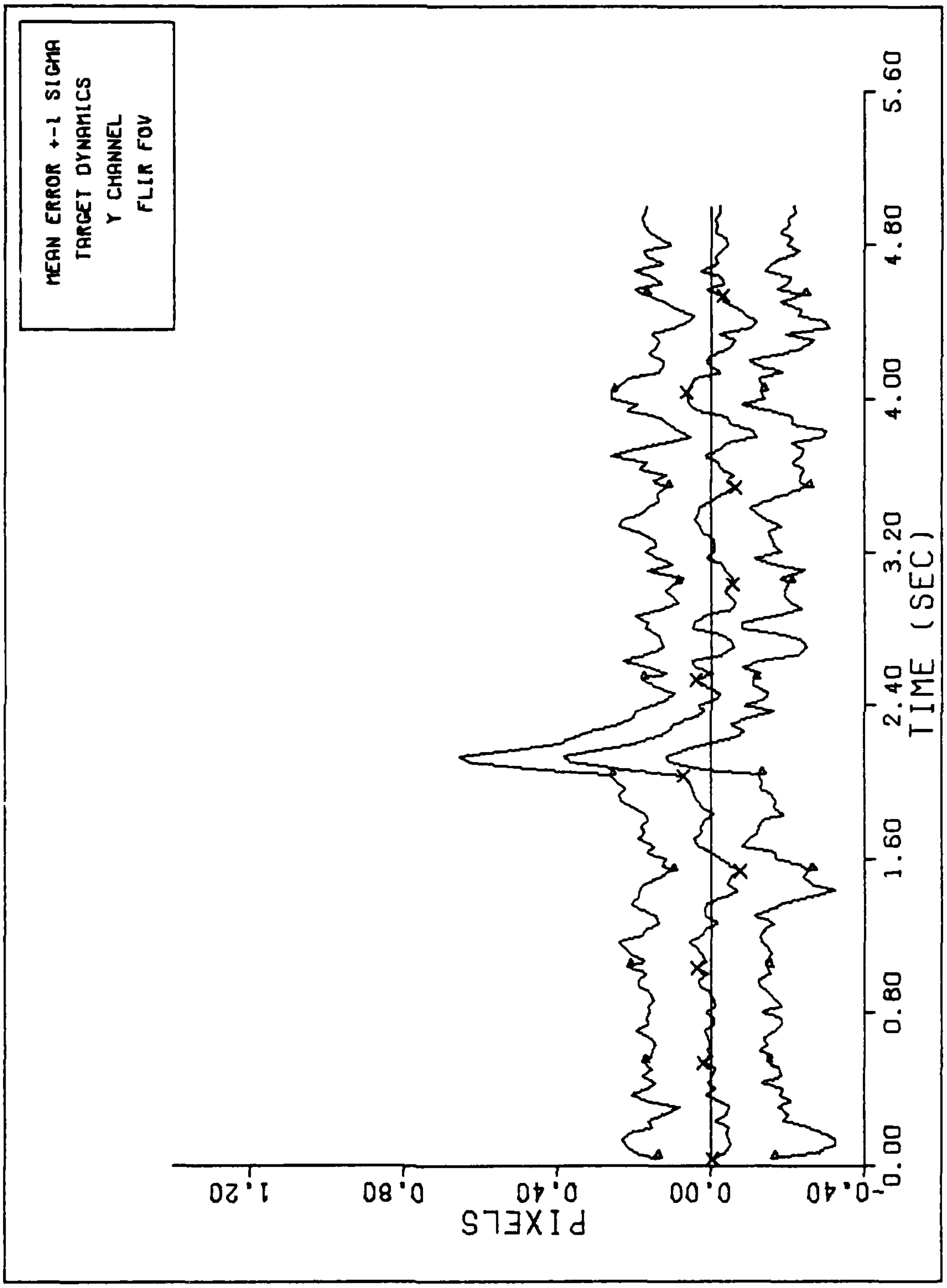
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-6c



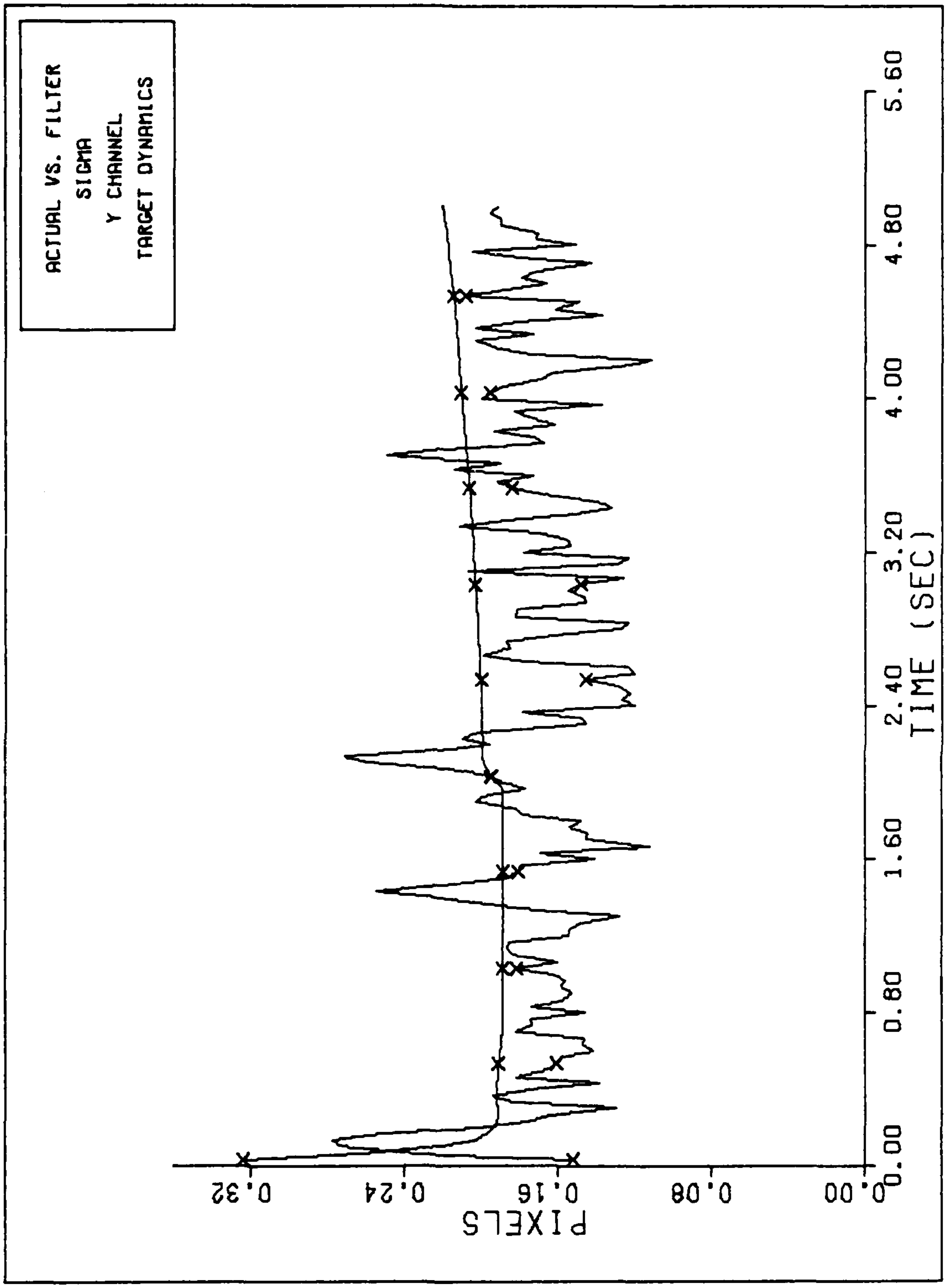
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-6d



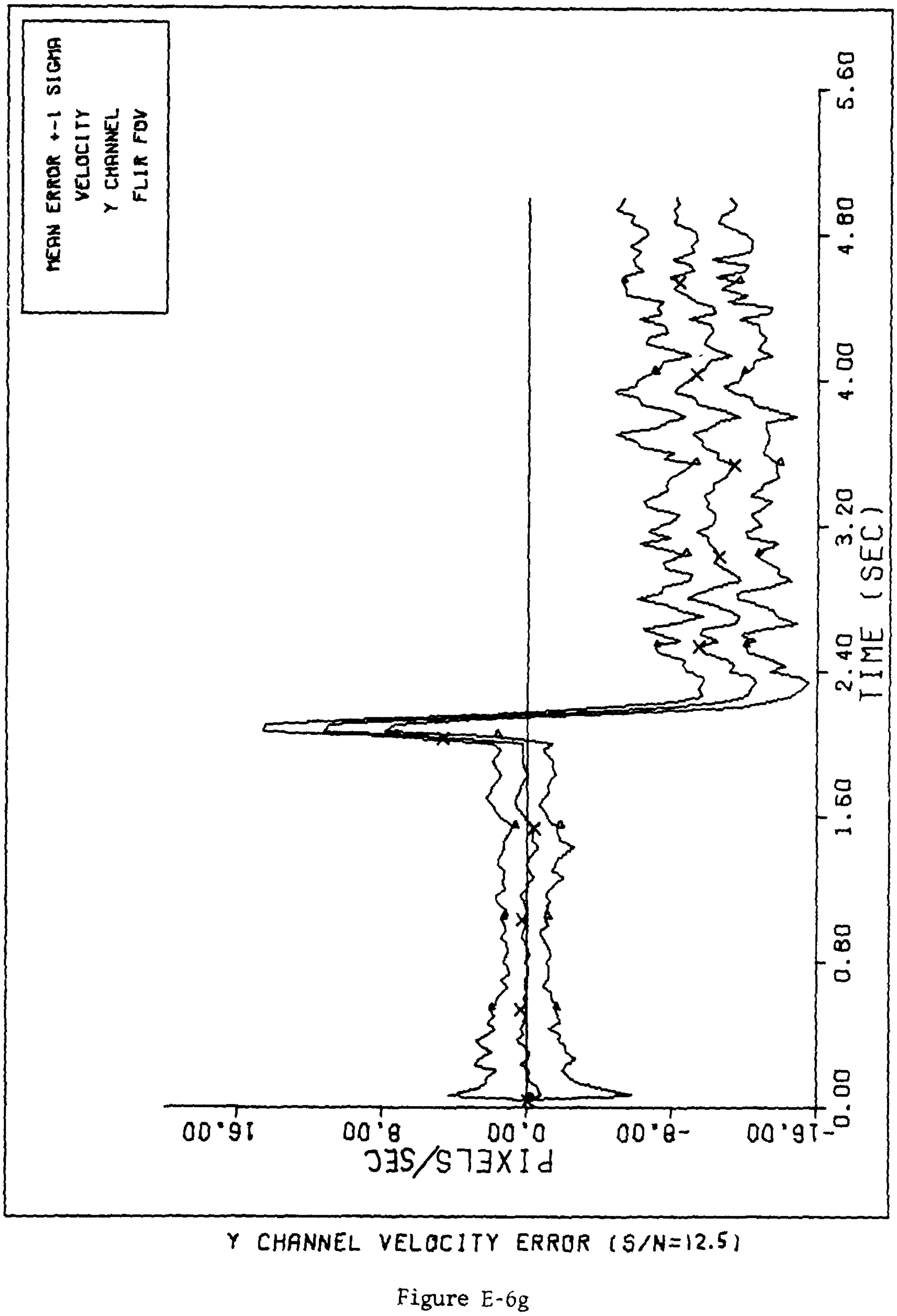
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-6e



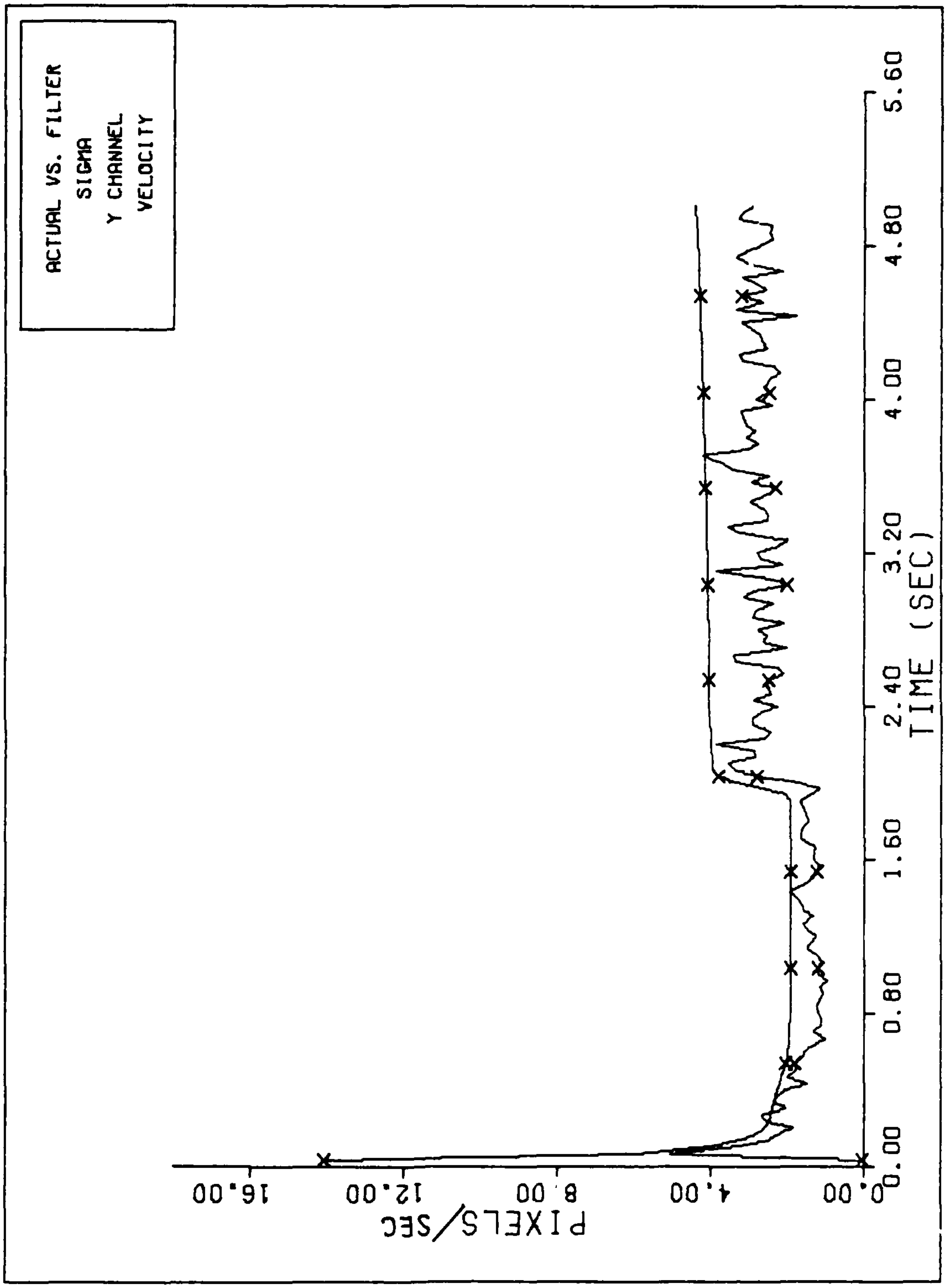
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-6f



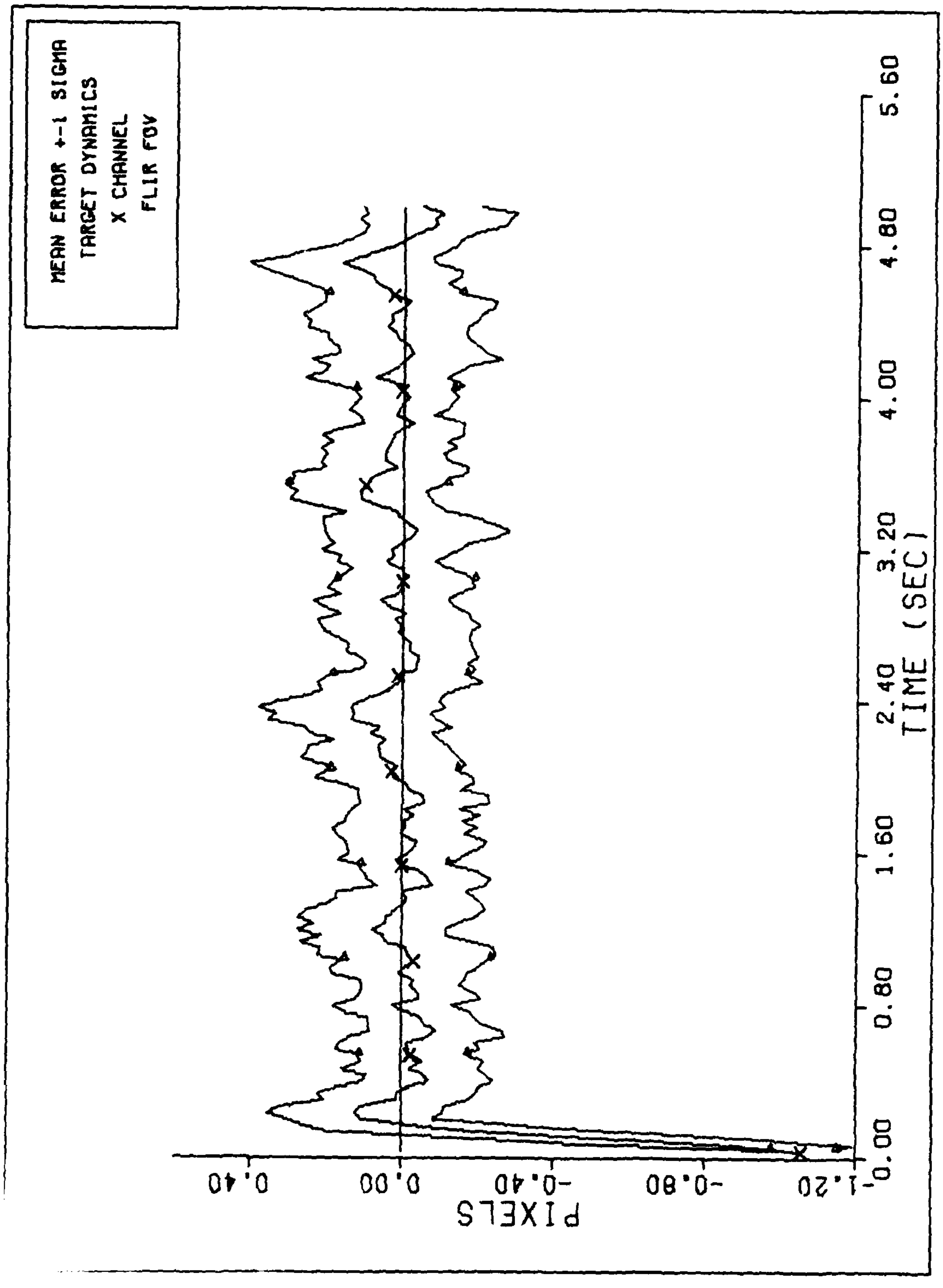
Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure E-6g



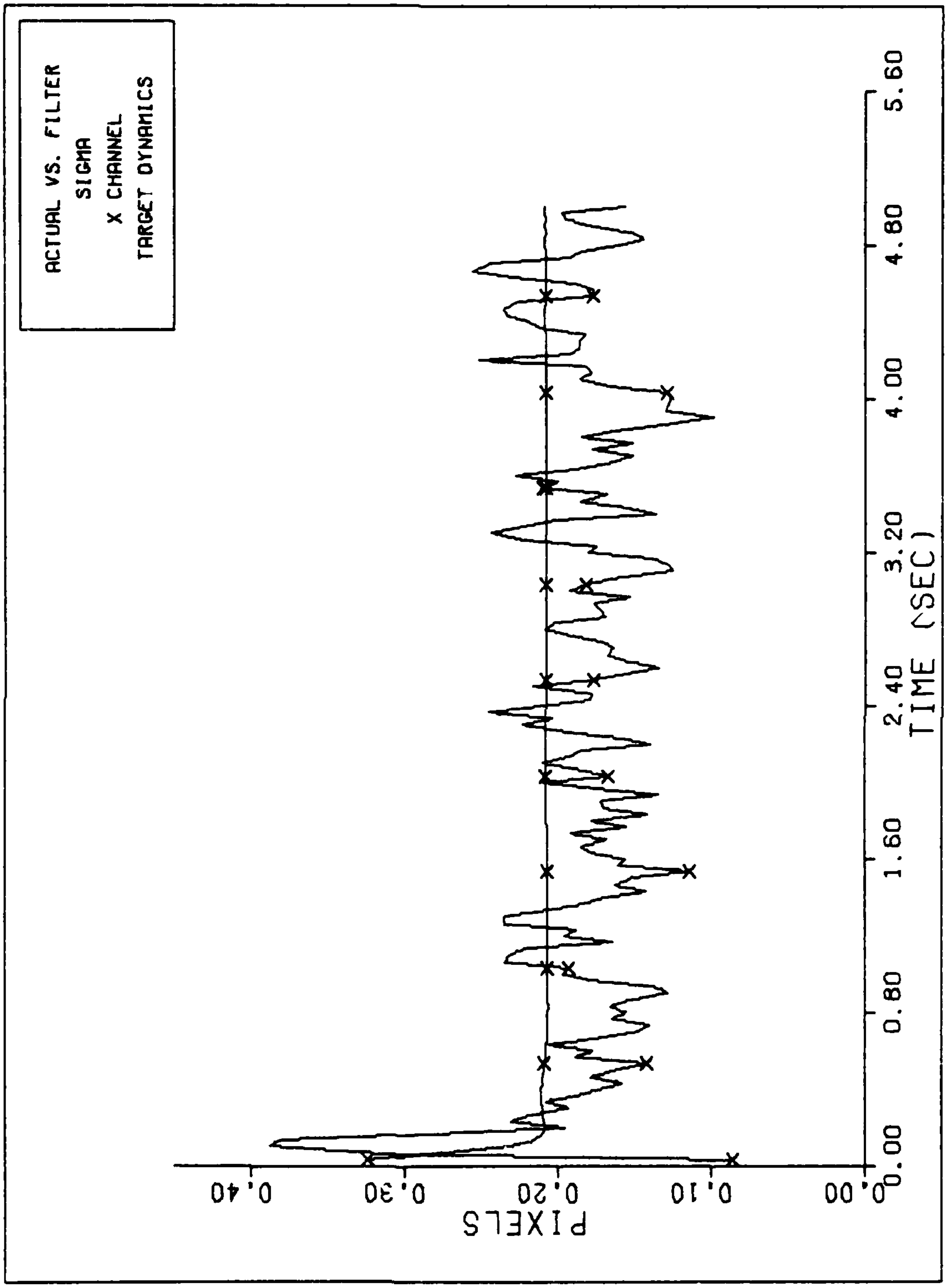
FILTER VS. ACTUAL SIGMA PLOT (S/N = 12.5)

Figure E-6h



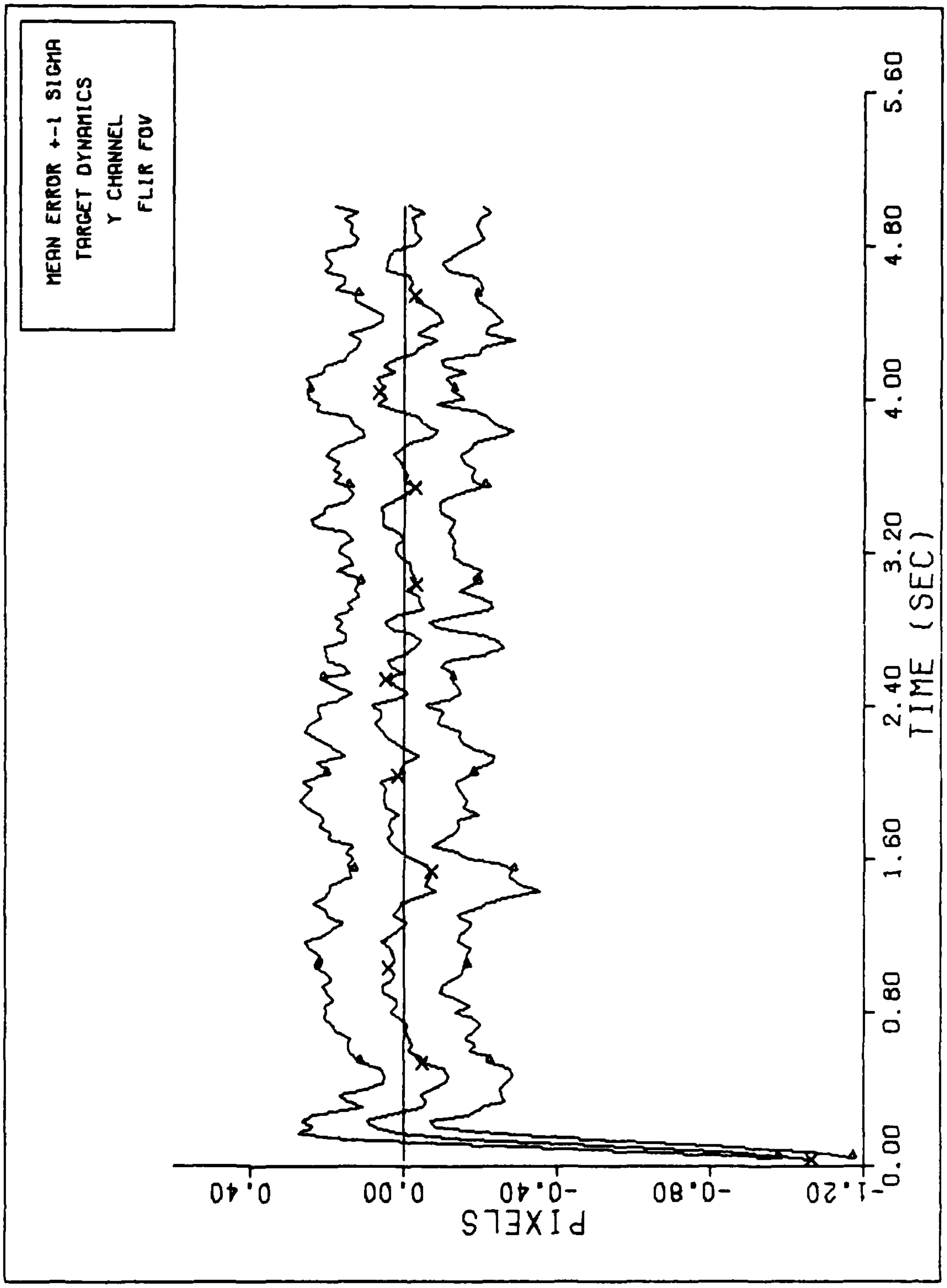
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-7a



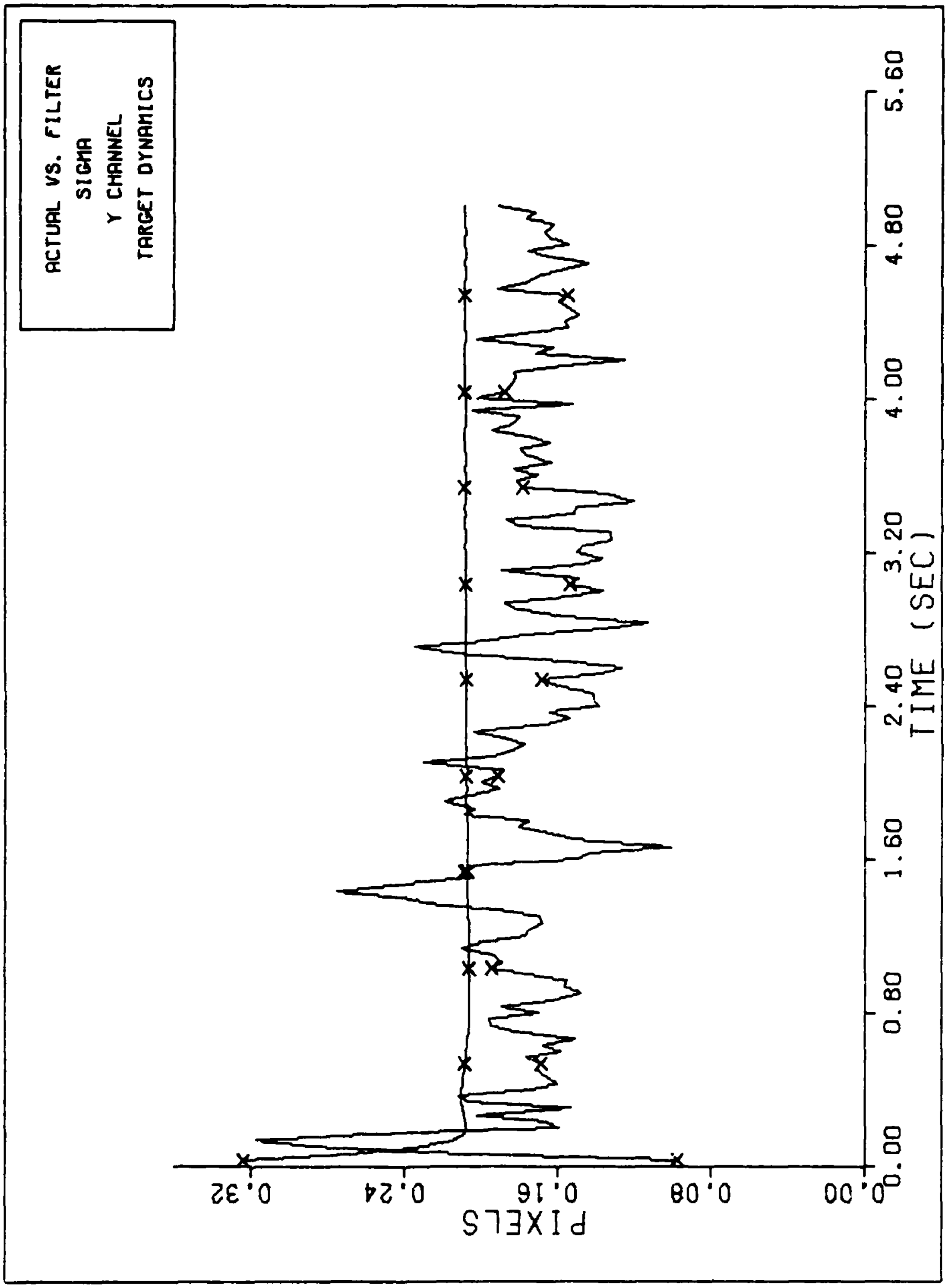
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-7b



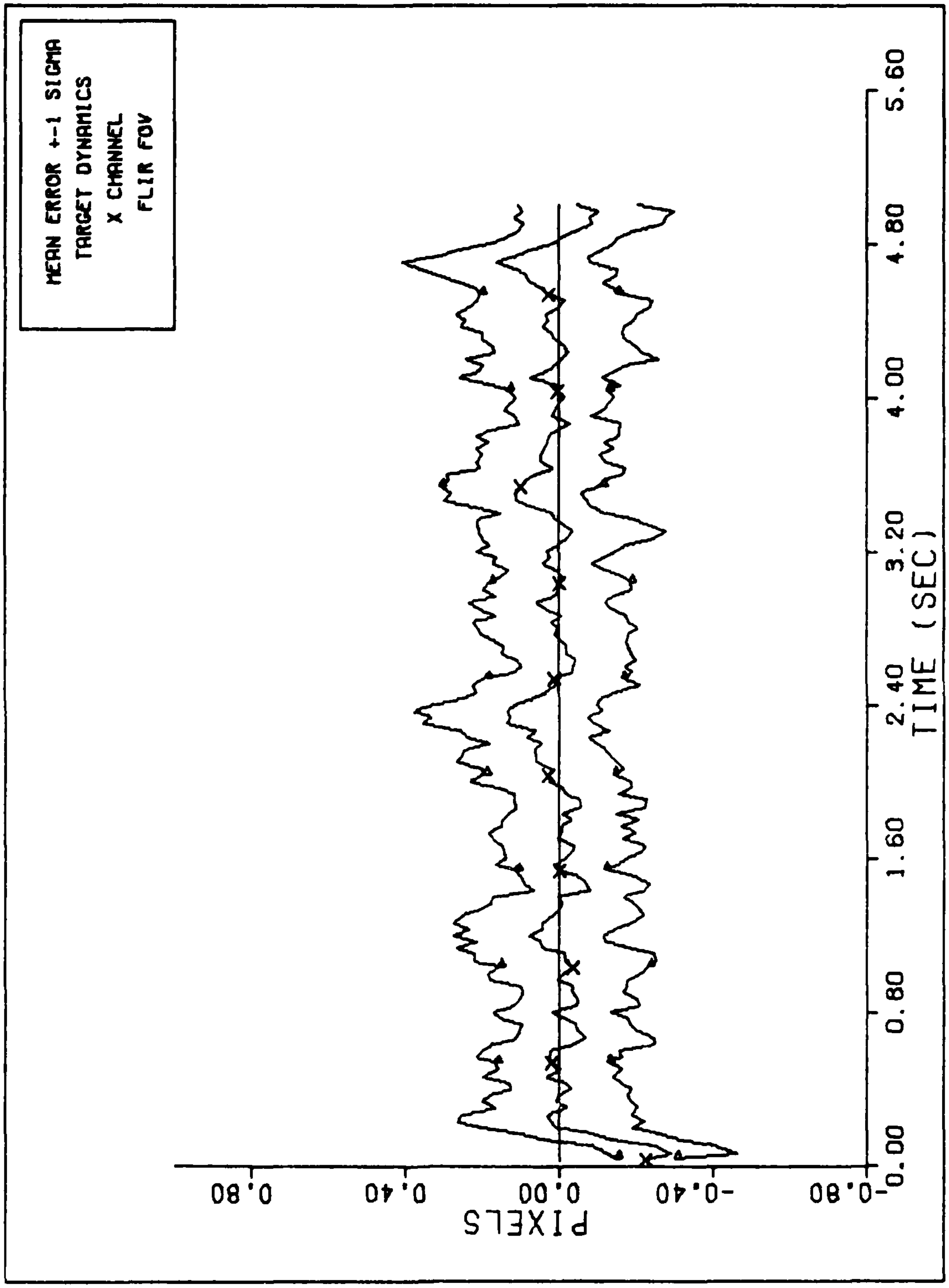
Y CHANNEL DYNAMICS ERROR [S/N=12.5]

Figure E-7c



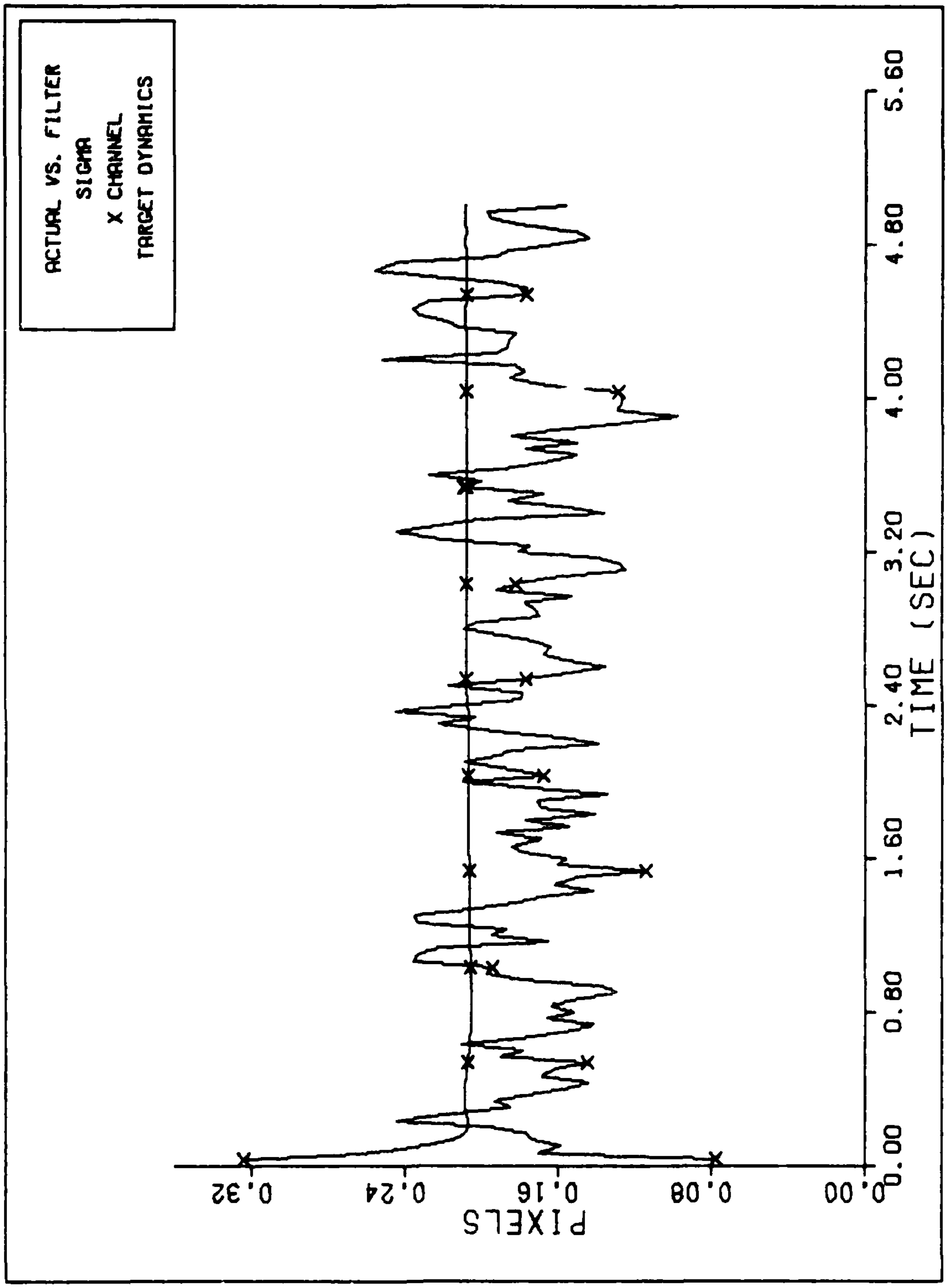
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-7d



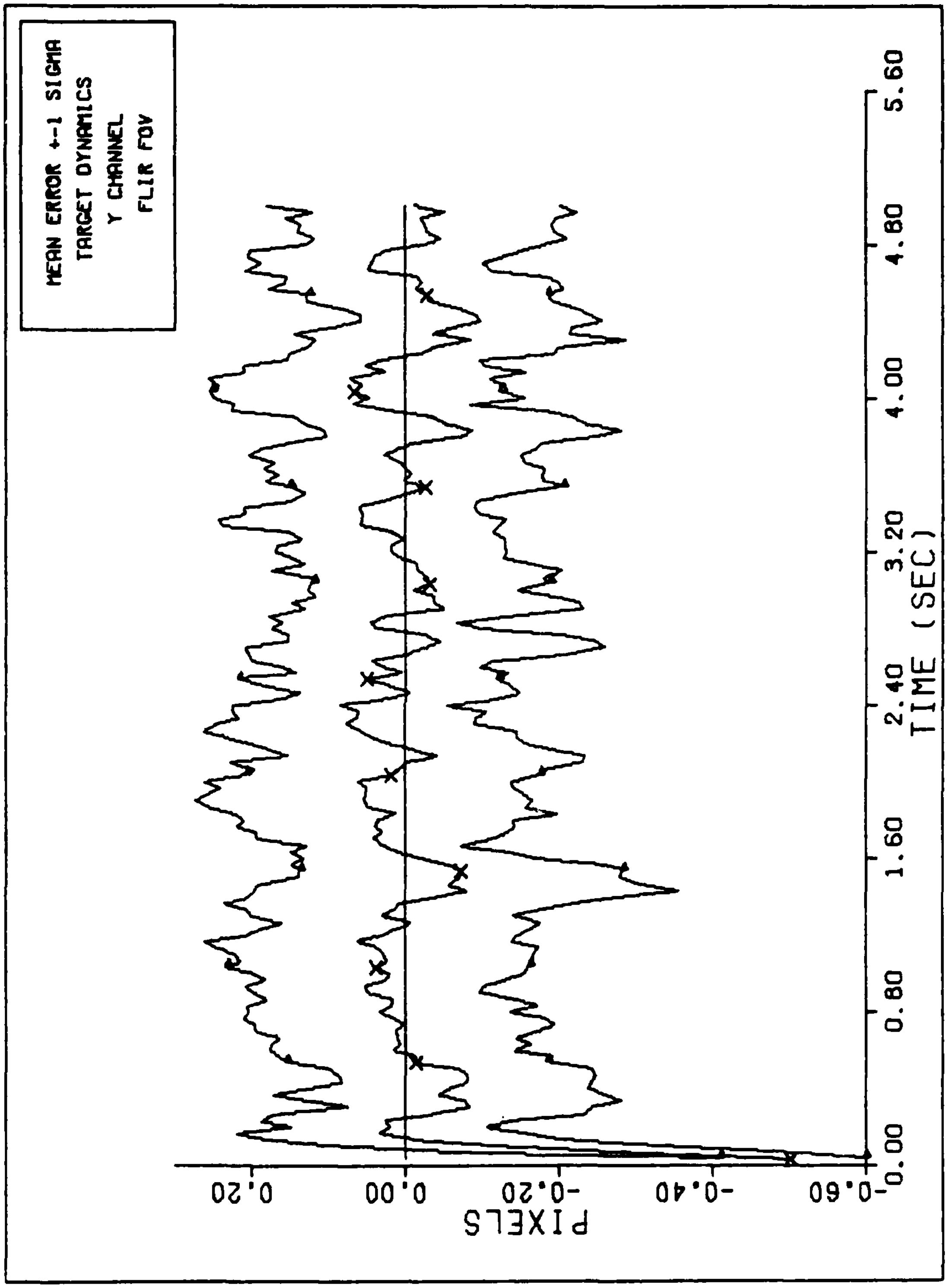
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-9a



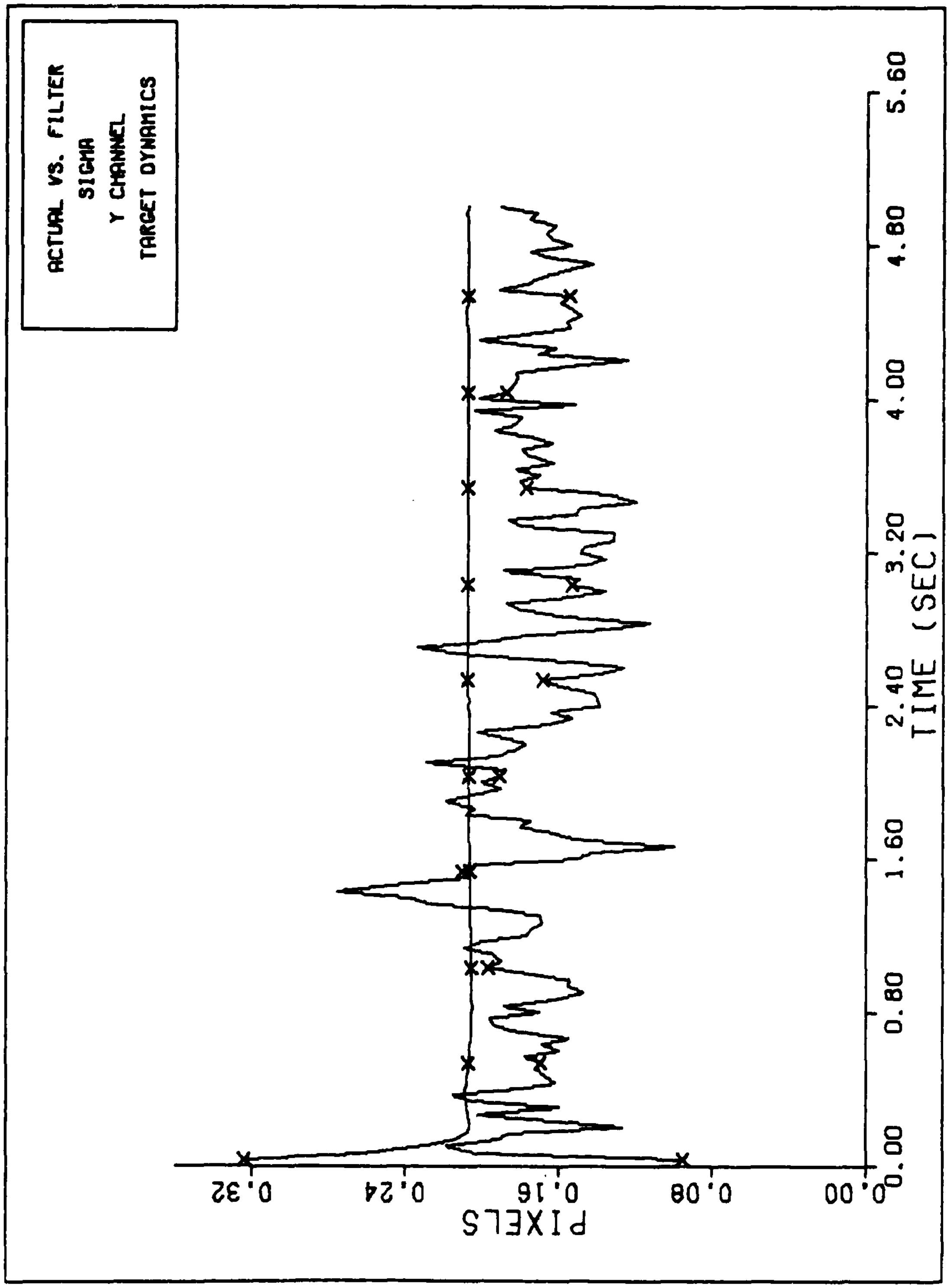
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-9b



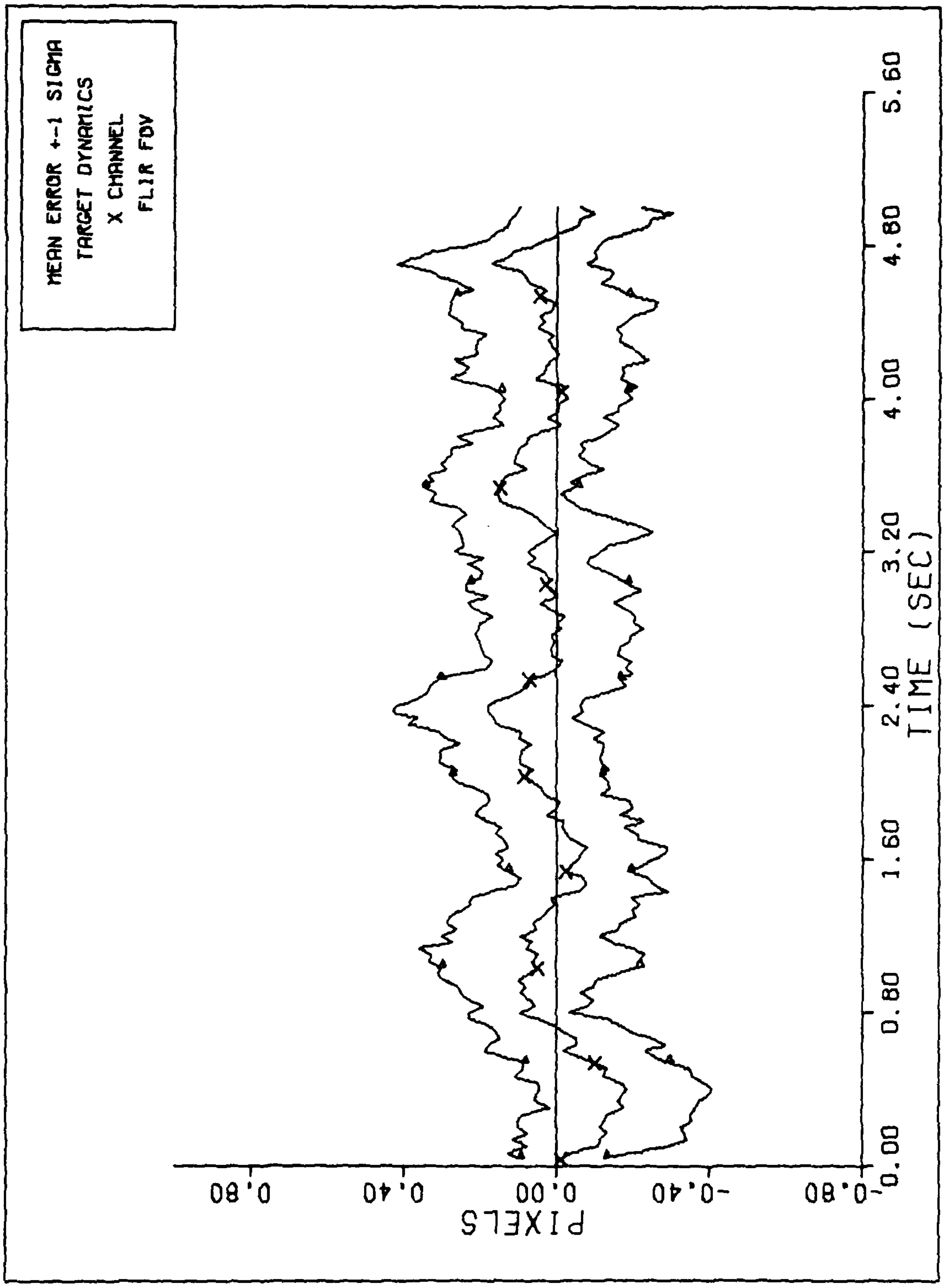
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure E-9c



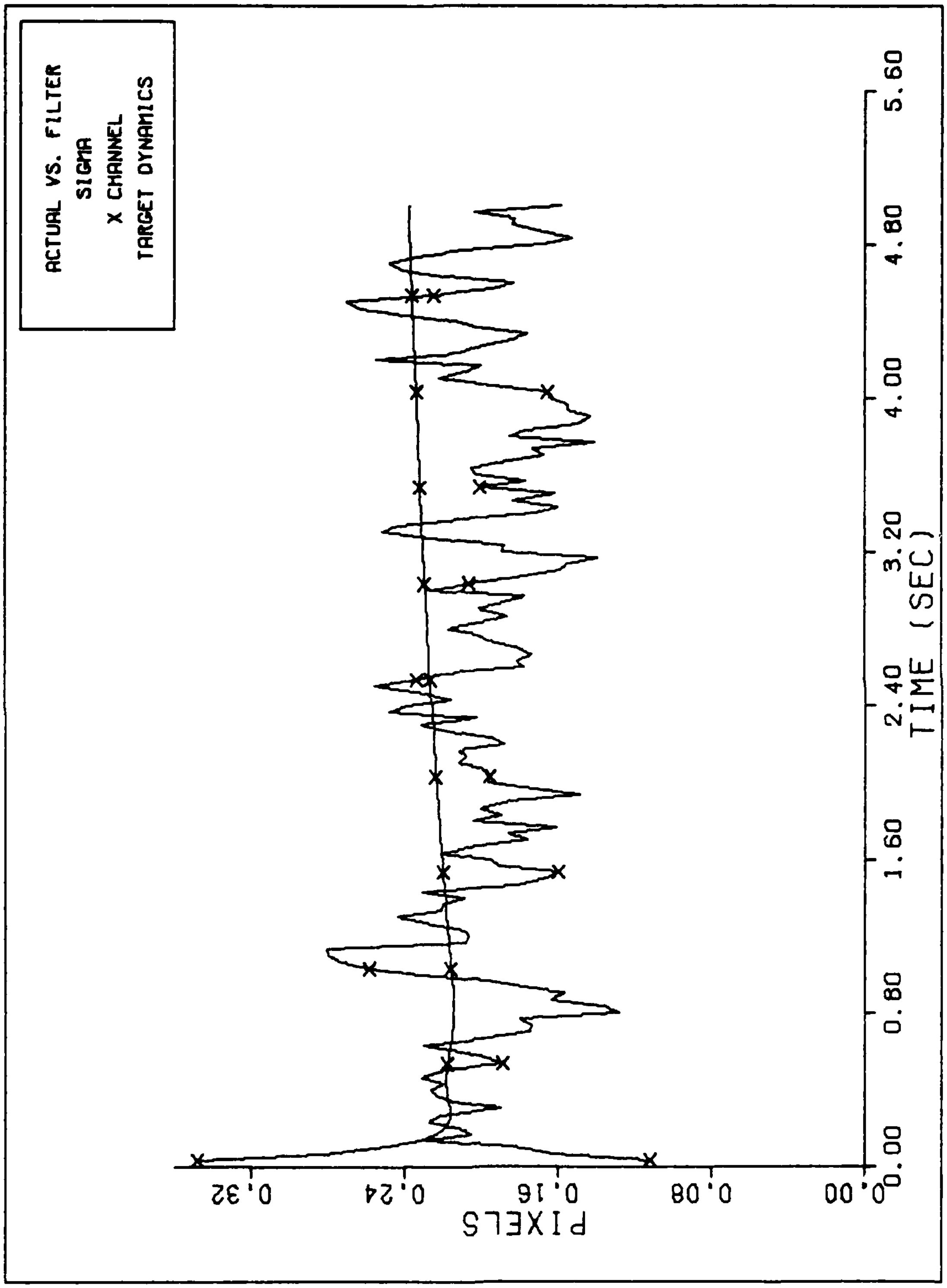
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure E-9d



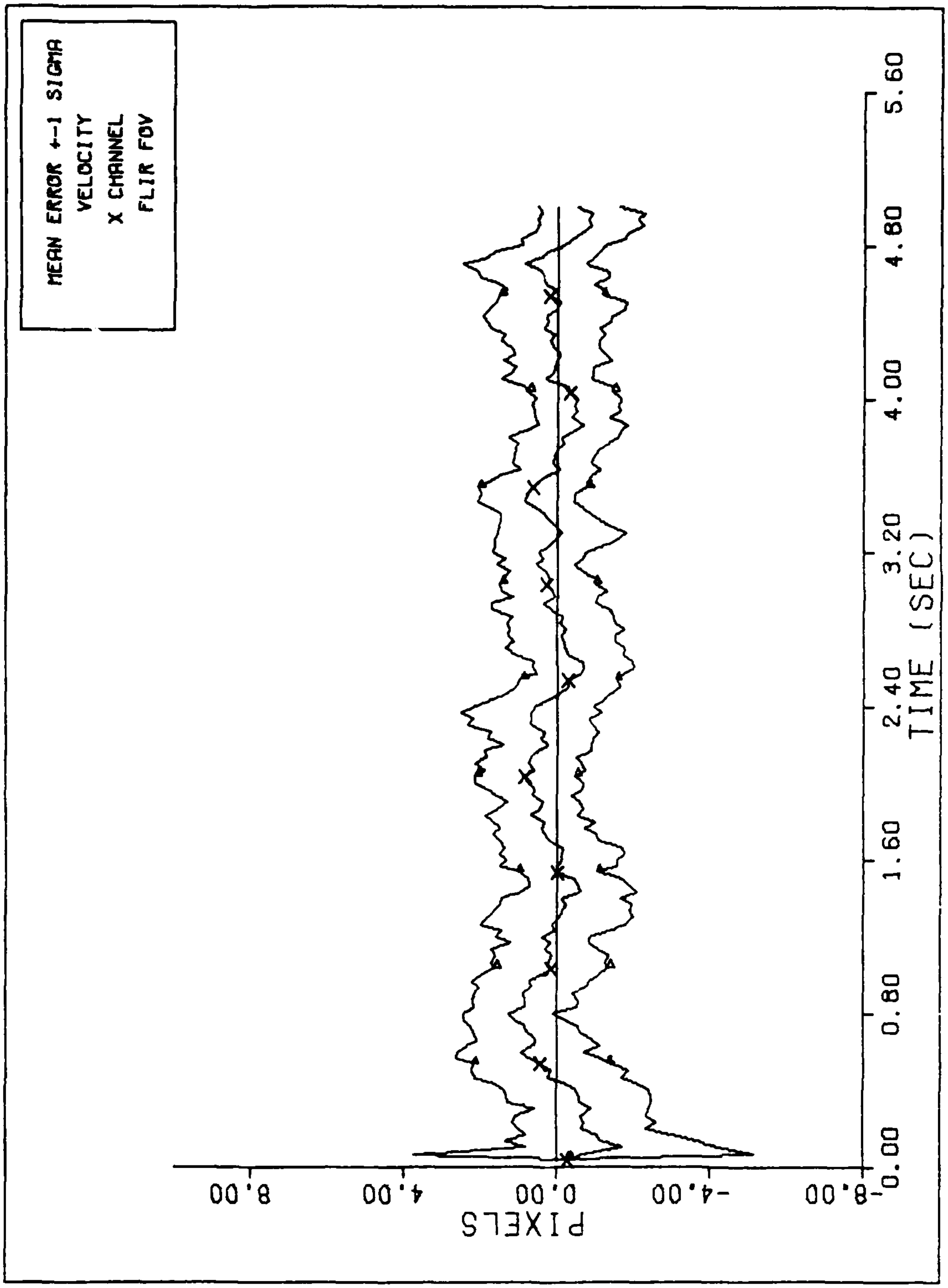
X CHANNEL DYNAMICS ERROR (S/N= 5)

Figure E-10a



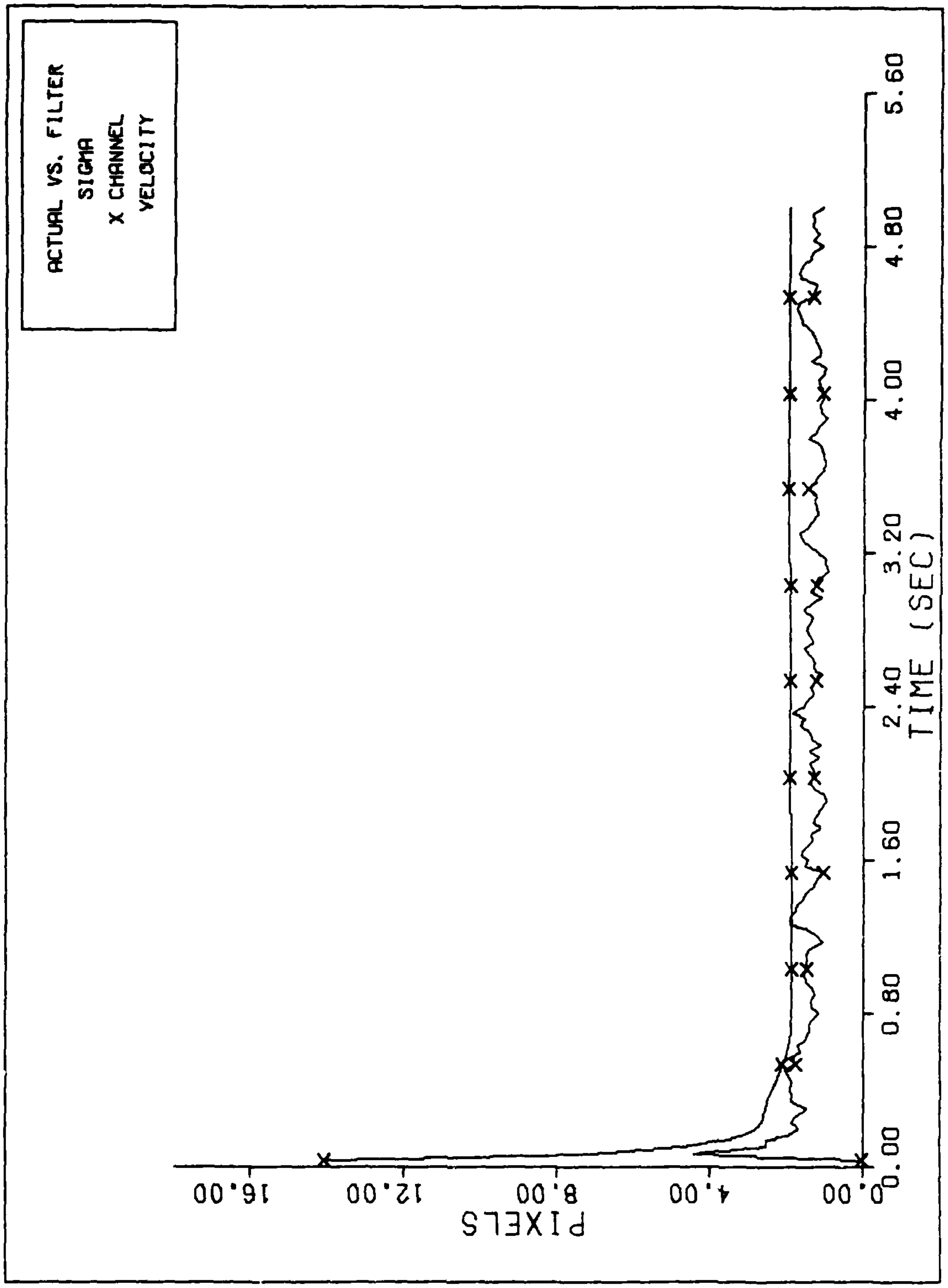
FILTER VS. ACTUAL SIGMA PLOT (S/N = 5)

Figure E-10b



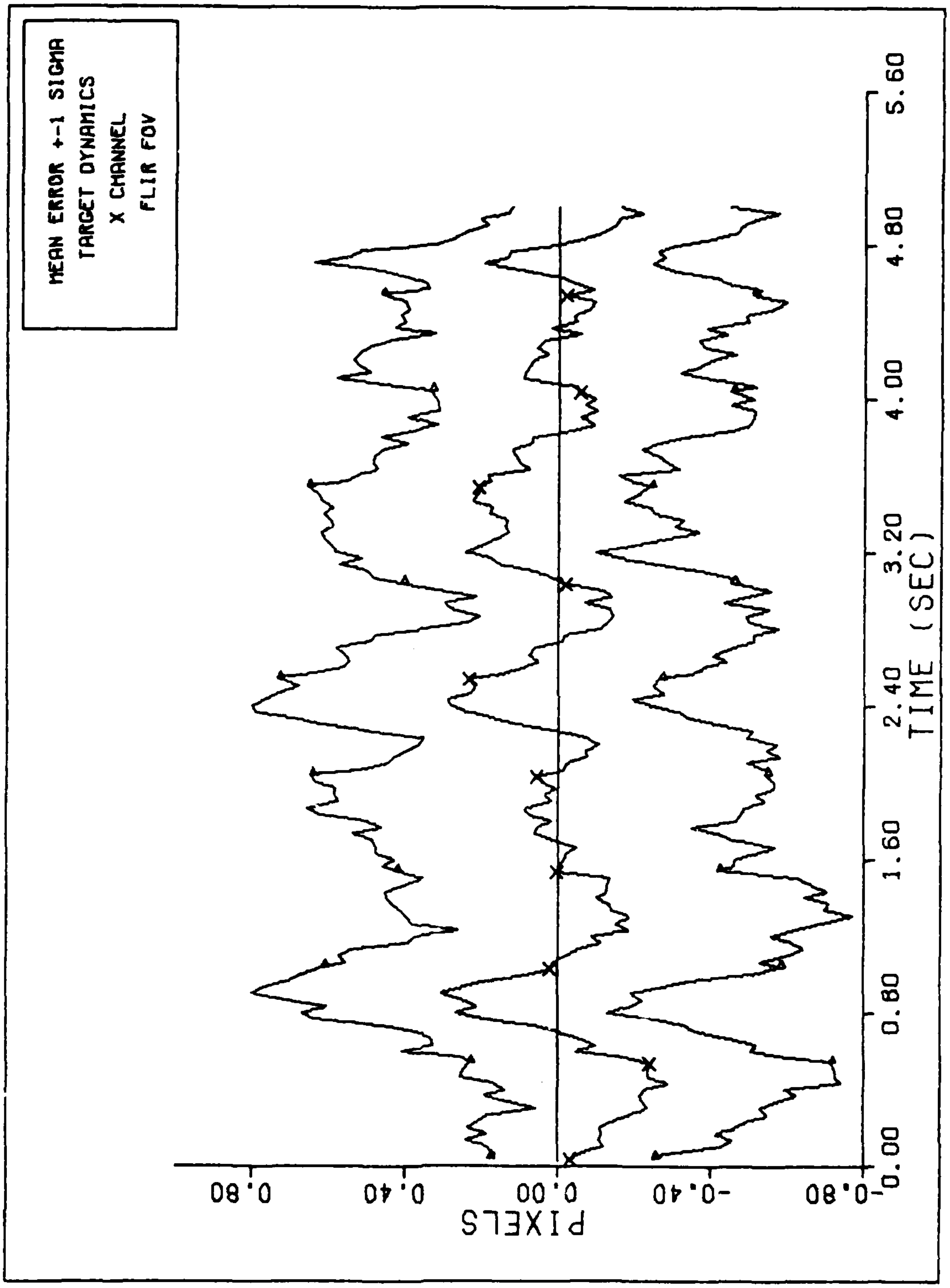
X CHANNEL VELOCITY ERROR (S/N= 5)

Figure E-10c



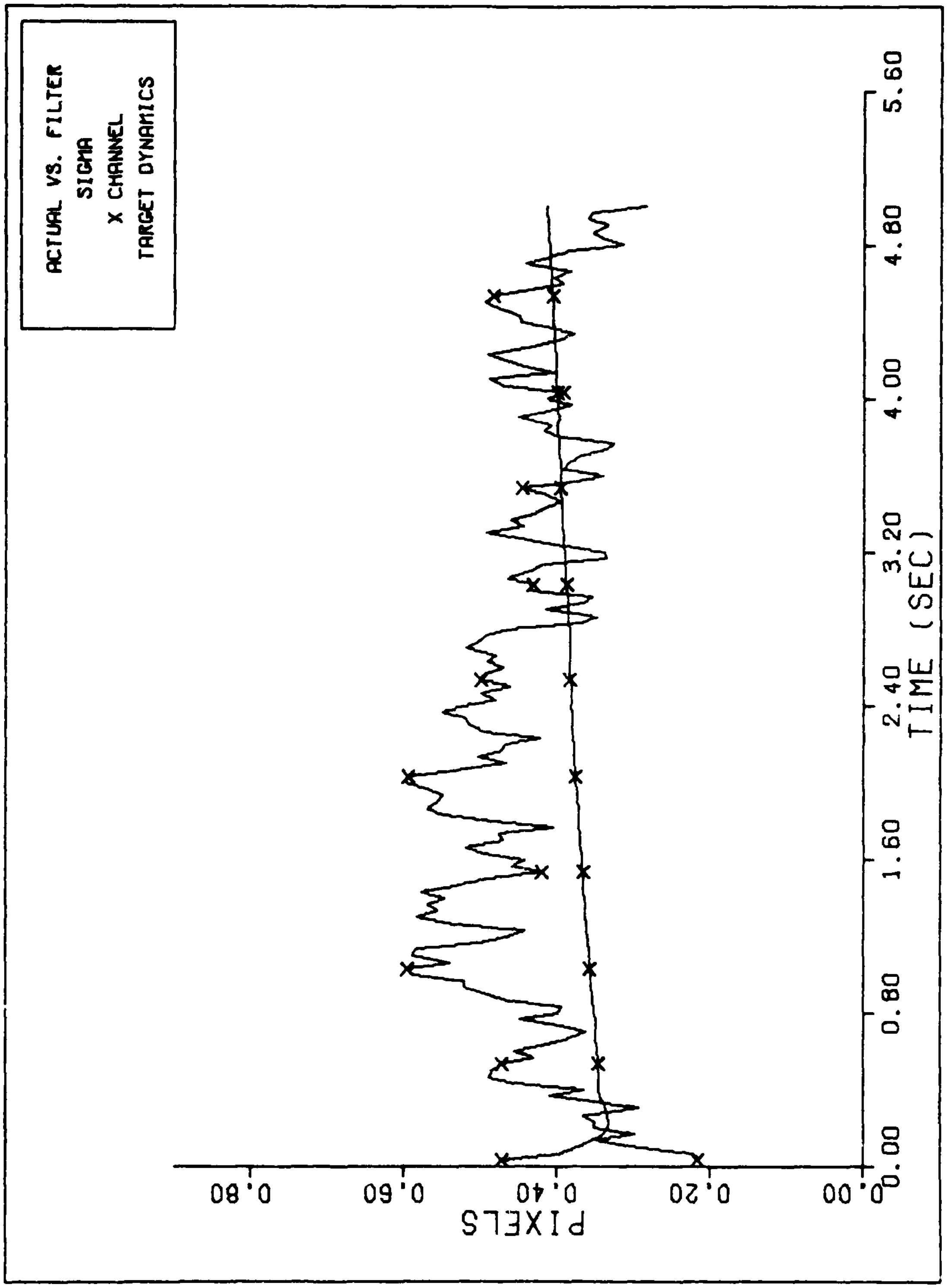
FILTER VS. ACTUAL SIGMA PLOT (S/N = 5)

Figure E-10d



X CHANNEL DYNAMICS ERROR (S/N= 2)

Figure E-11a



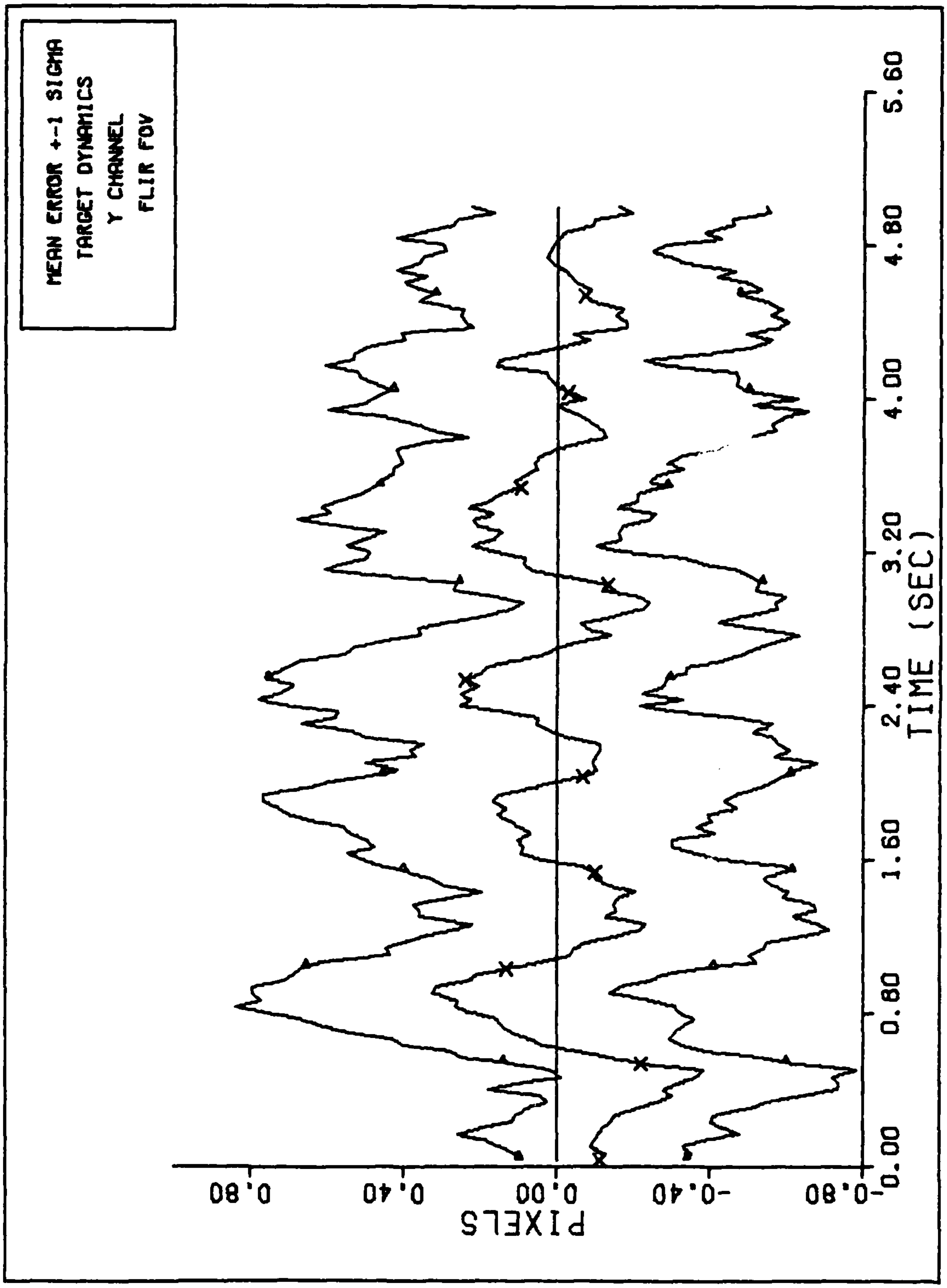
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure E-11b

AD-A111 144 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 17/5
ALTERNATIVES TO AN EXTENDED KALMAN FILTER FOR TARGET IMAGE TRAC--ETC(U)
DEC 81 P R LEUTHAUSER
UNCLASSIFIED AFIT/GA/AA/81D-8 NL

3 OF 3
AD
A111144

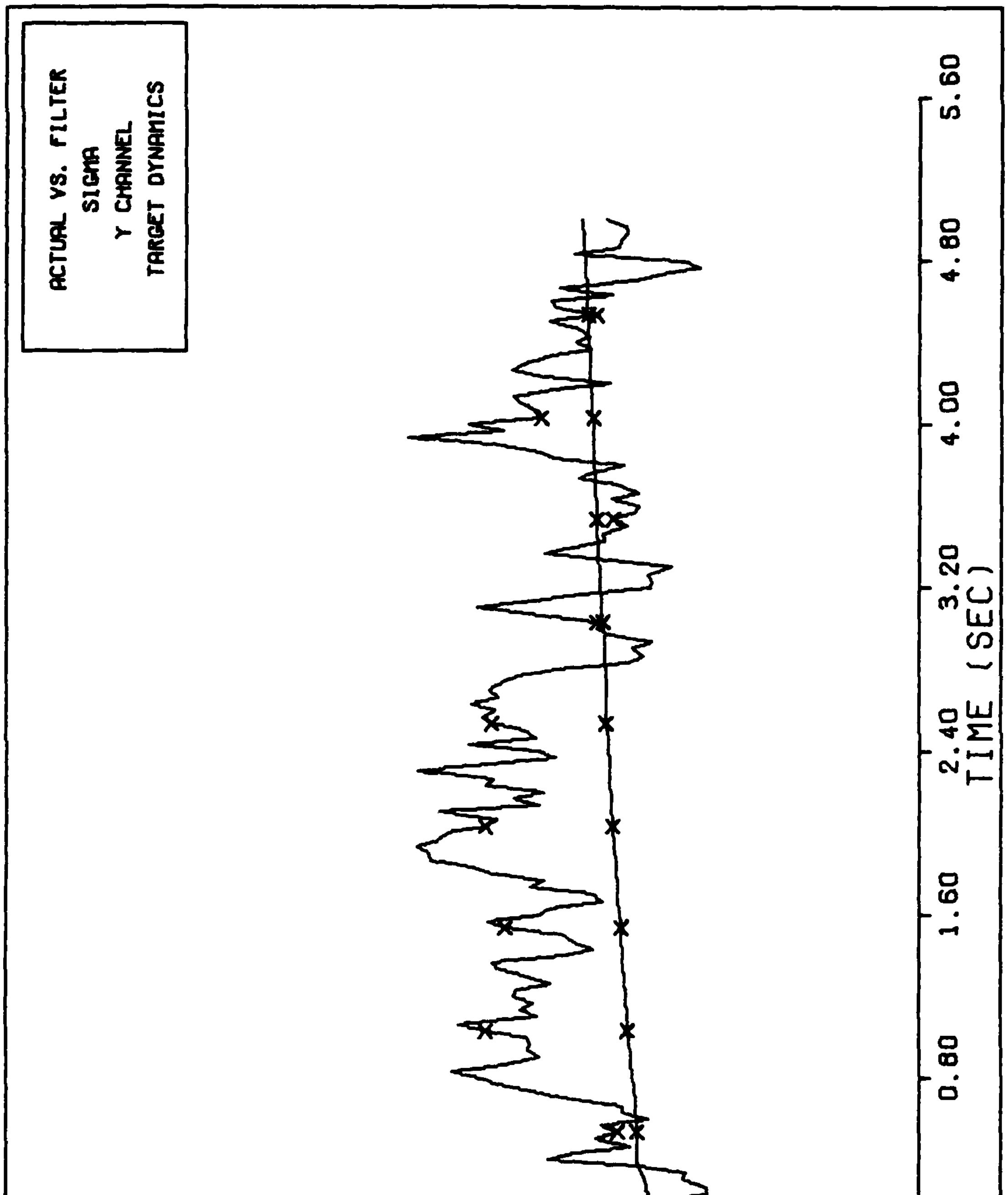
END
DATE
FILMED
3-82
DTIC



Y CHANNEL DYNAMICS ERROR (S/N= 2)

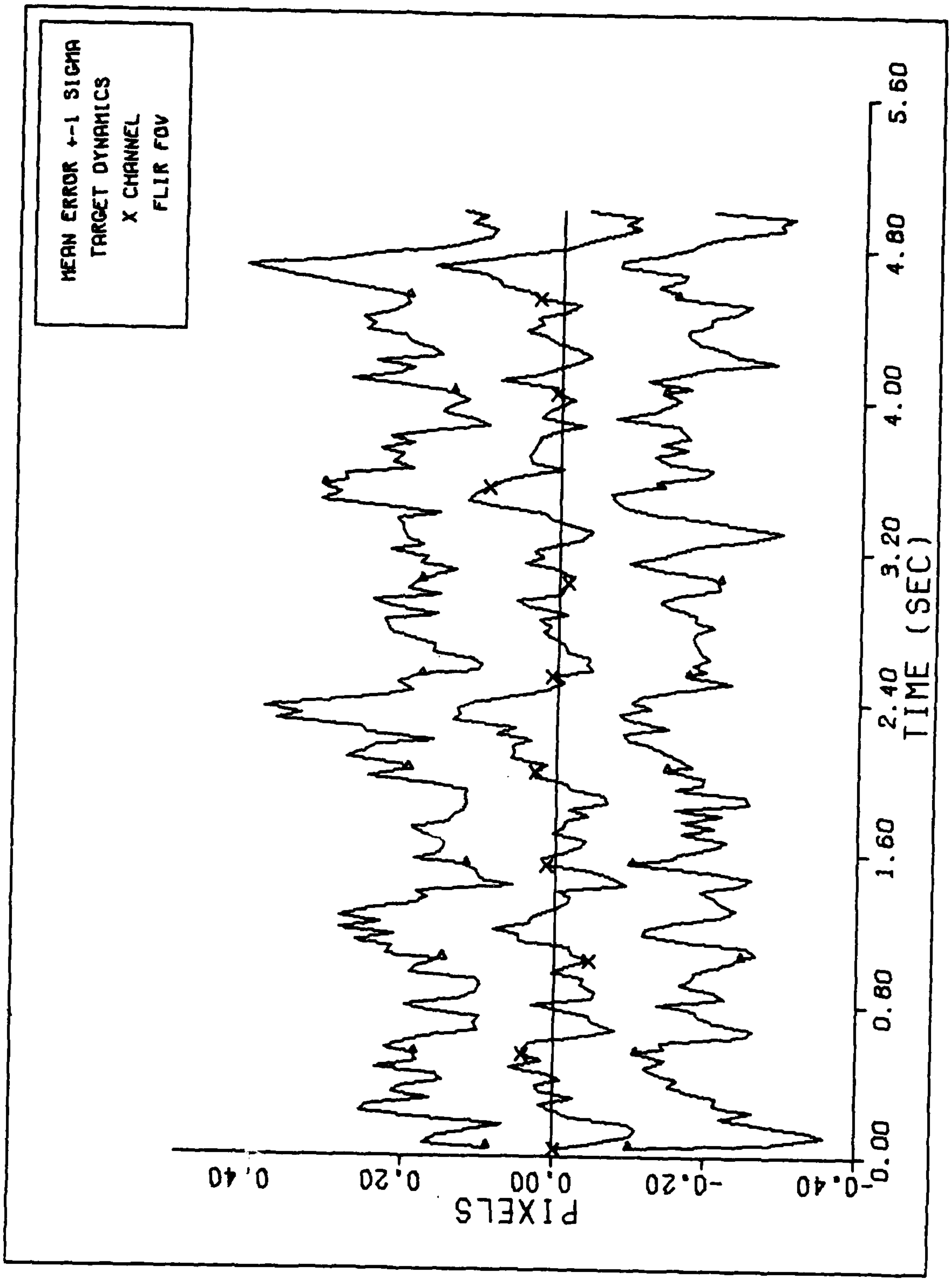
Figure E-11c

E11c



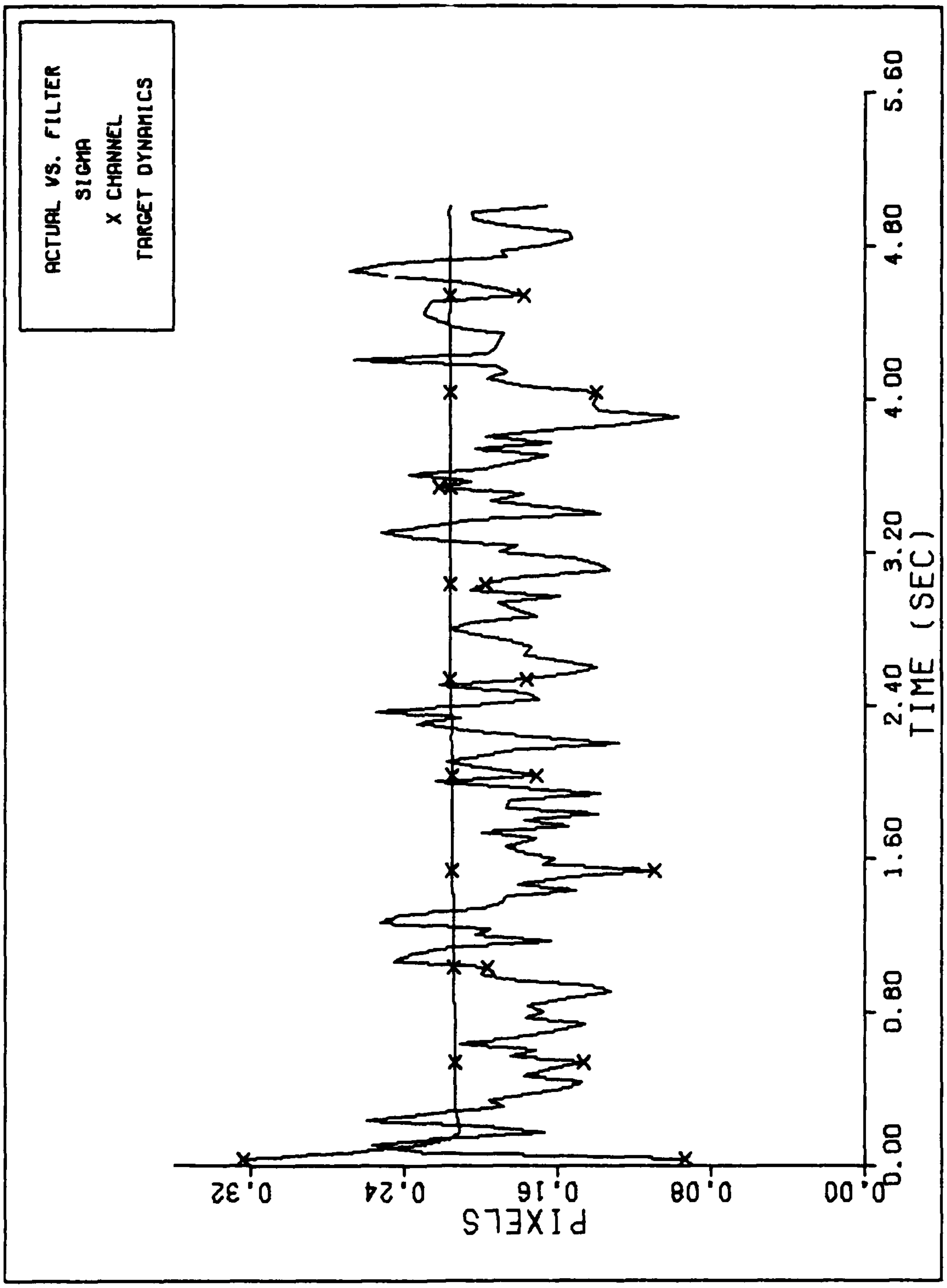
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure E-11d



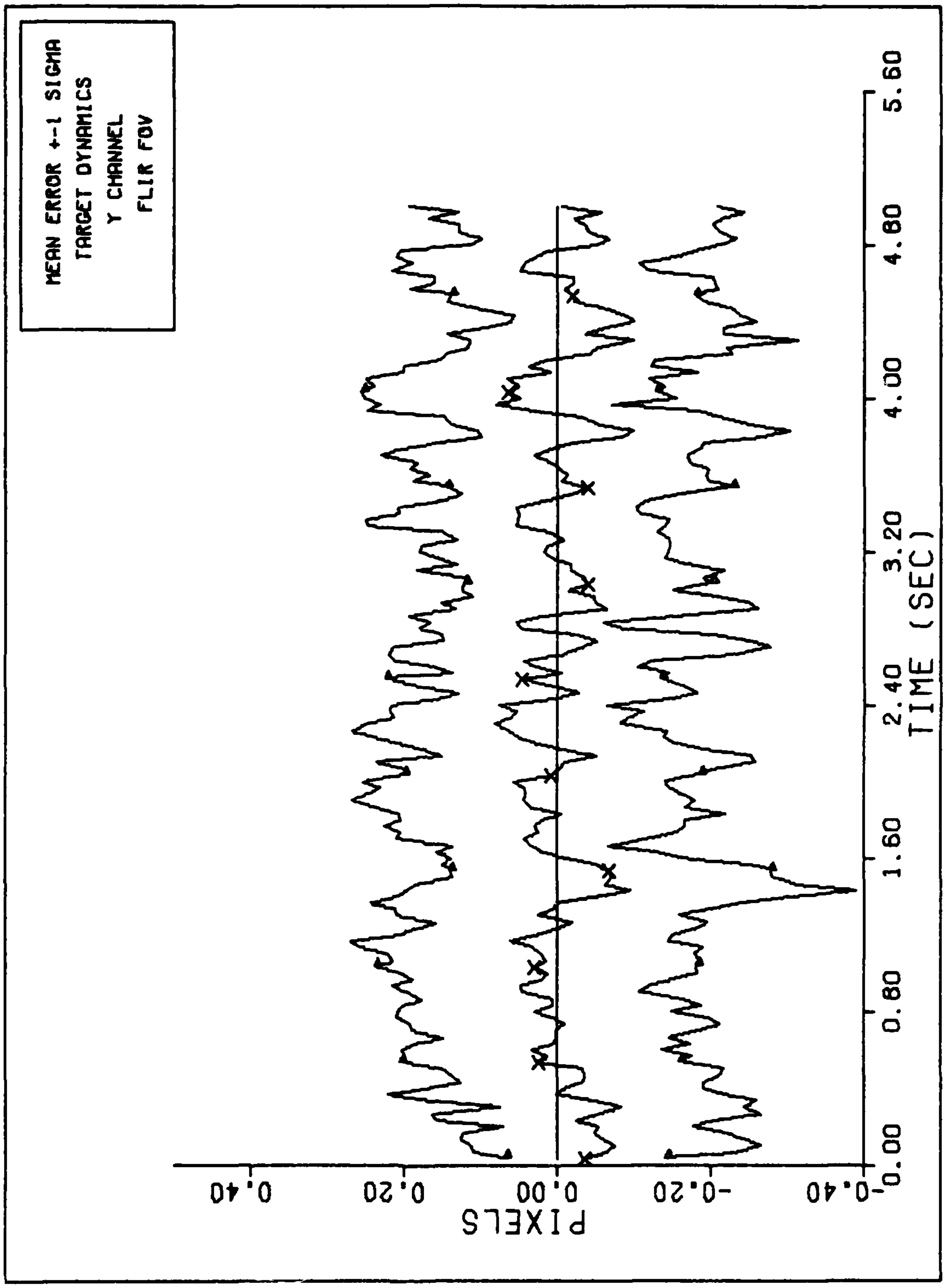
X CHANNEL DYNAMICS ERROR (S/N= 2)

Figure E-12a



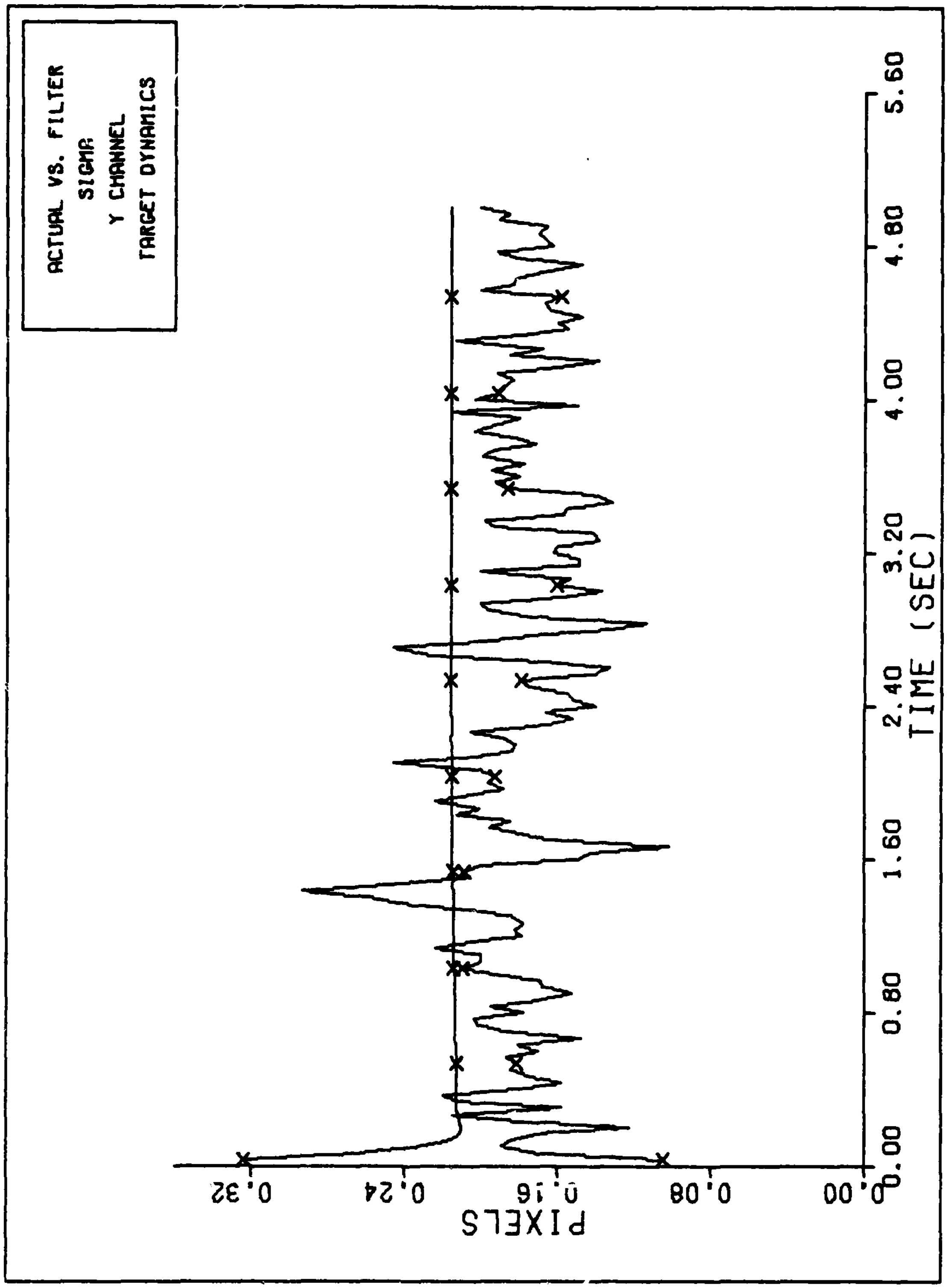
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure E-12b



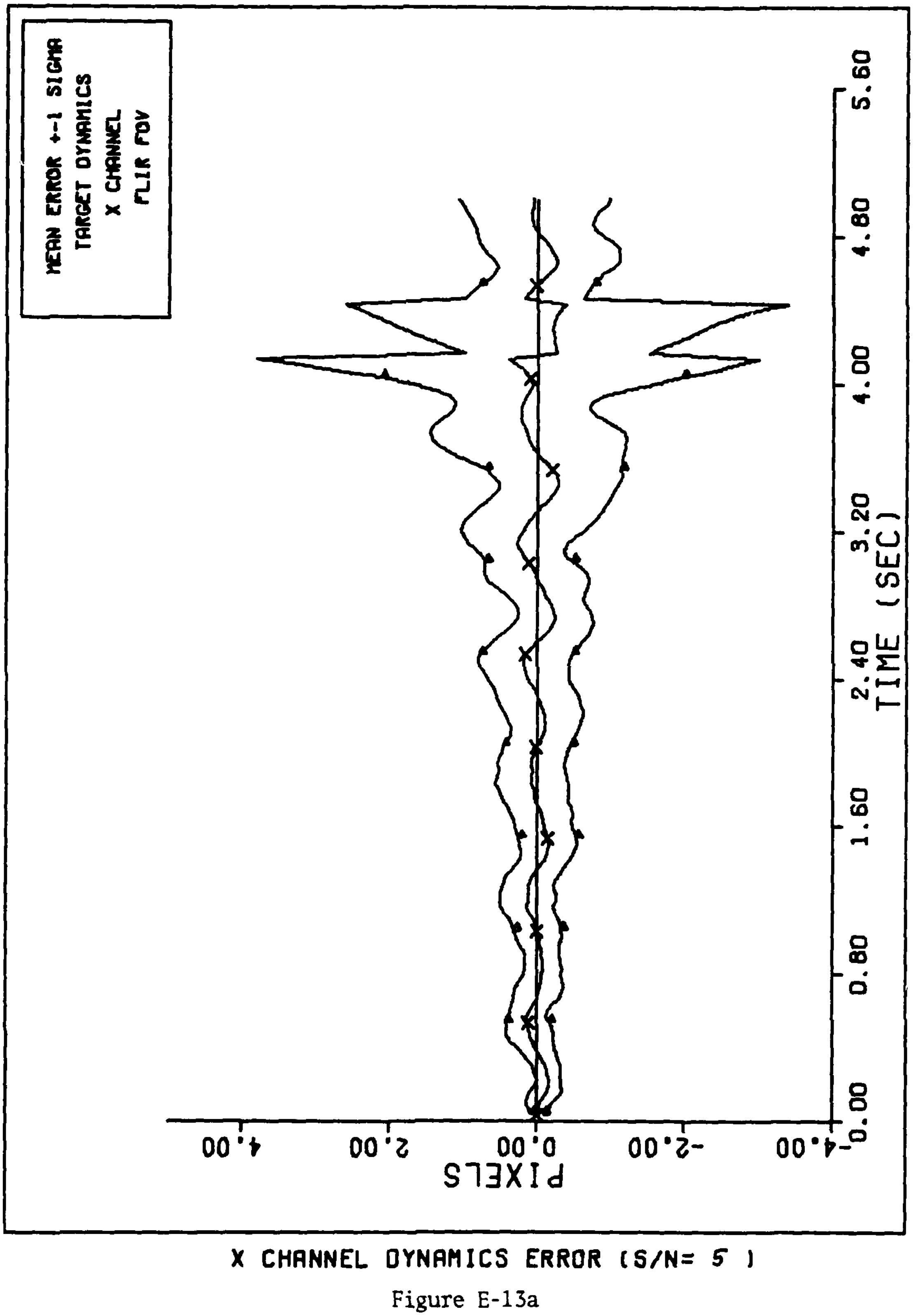
Y CHANNEL DYNAMICS ERROR (S/N= 2)

Figure E-12c



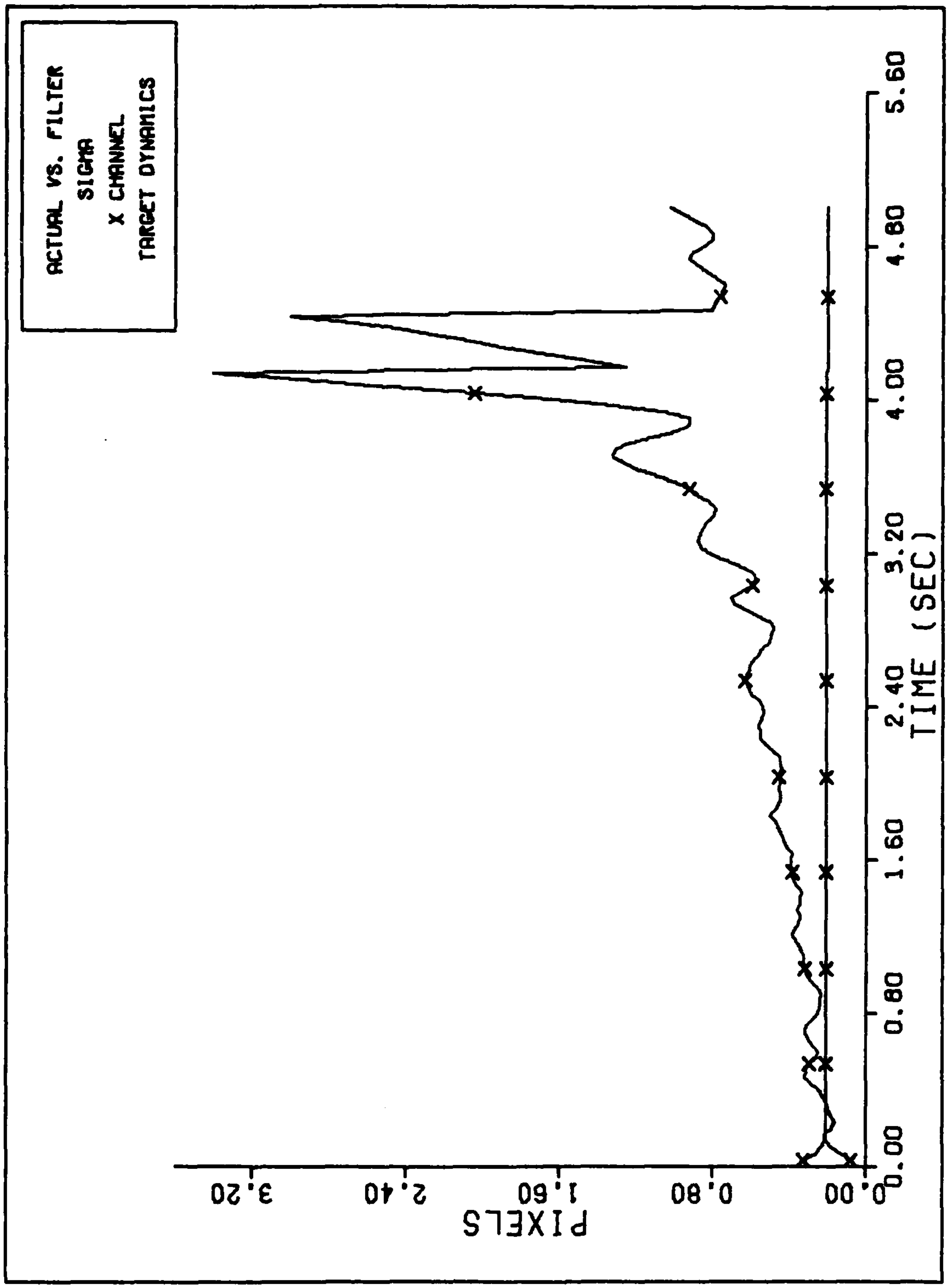
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure E-12d



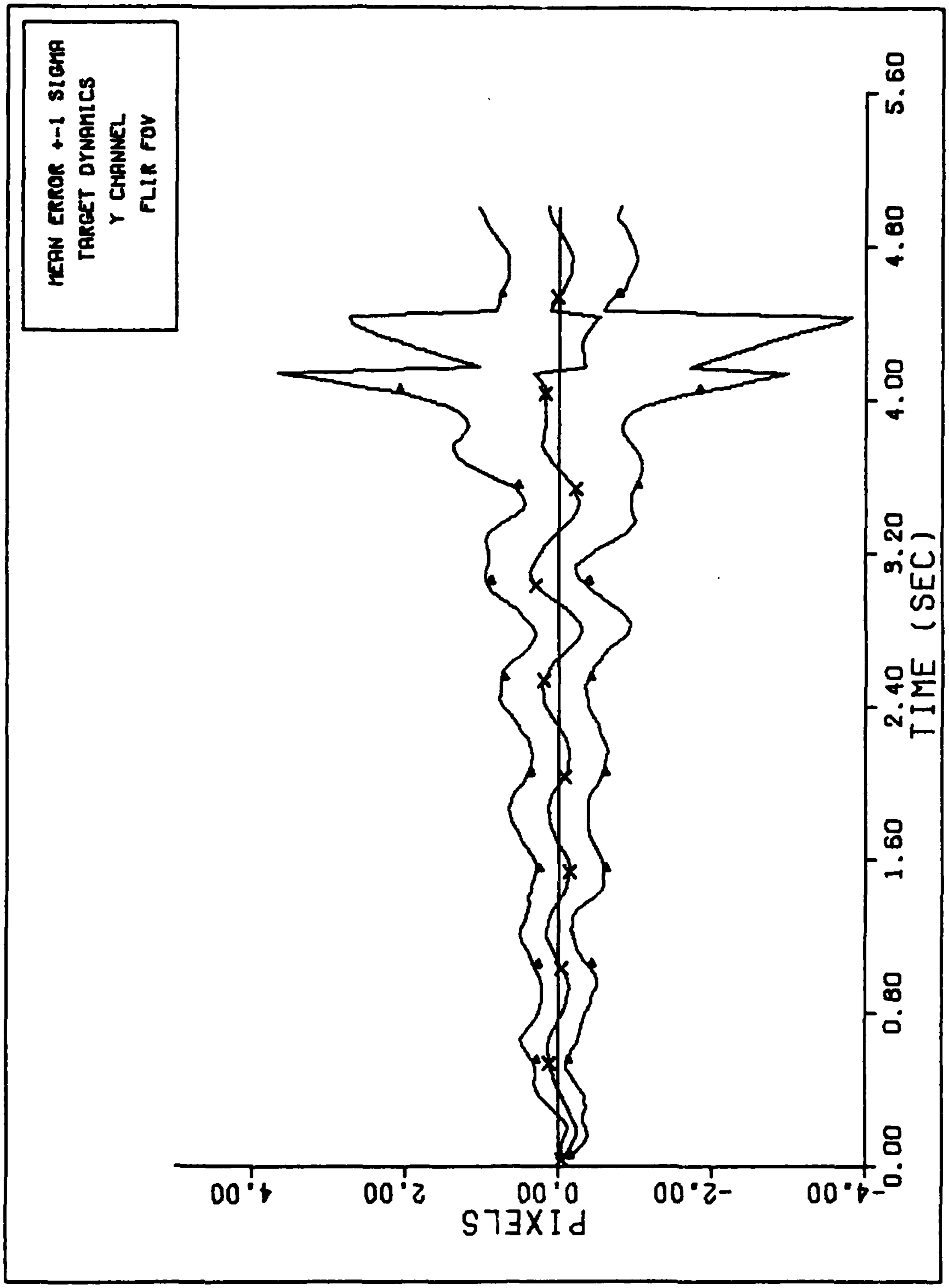
X CHANNEL DYNAMICS ERROR (S/N= 5)

Figure E-13a



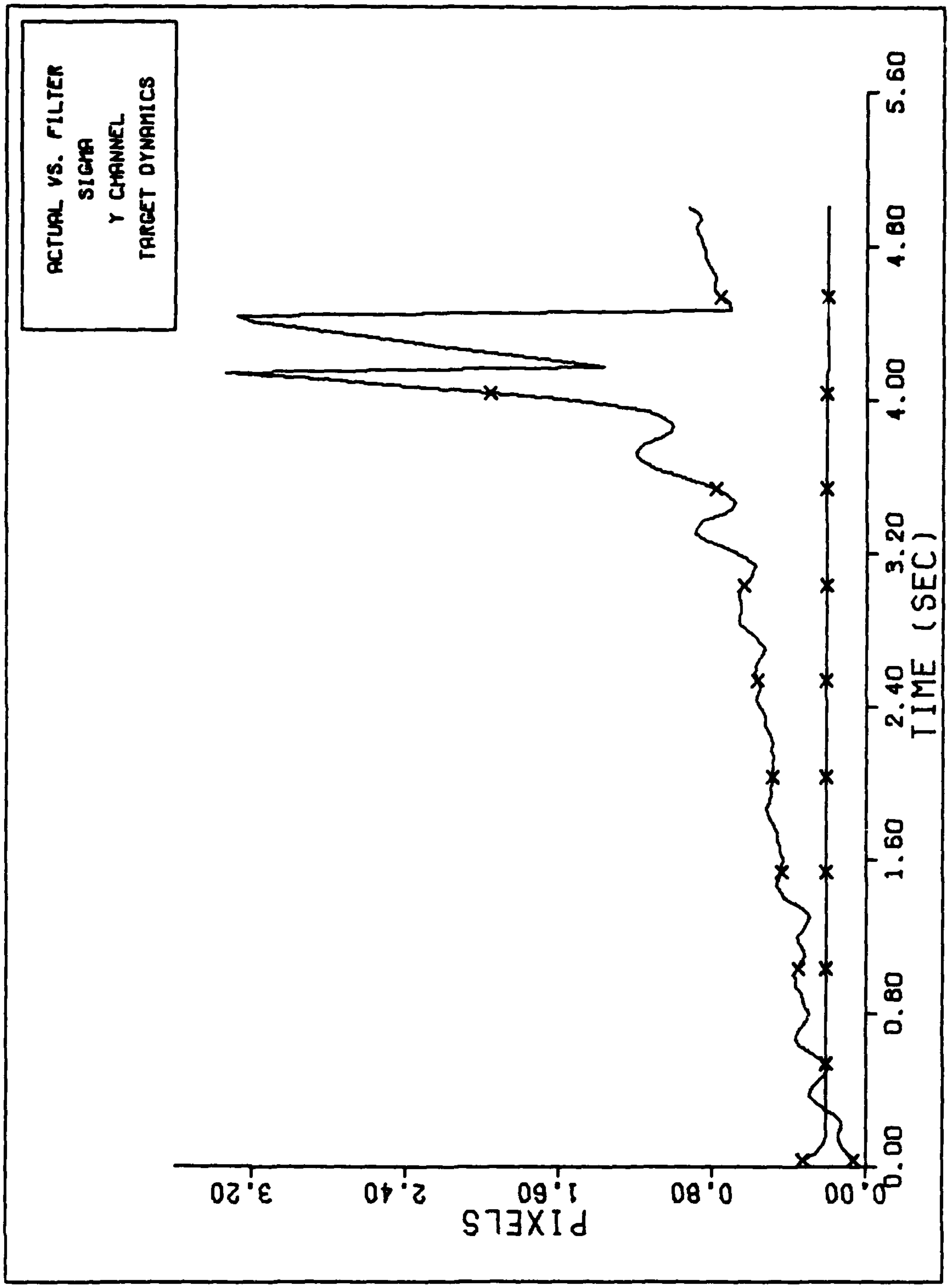
FILTER VS. ACTUAL SIGMA PLOT (S/N = 5)

Figure E-13b



Y CHANNEL DYNAMICS ERROR (S/N= 5)

Figure E-13c



FILTER VS. ACTUAL SIGMA PLOT (S/N = 5)

Figure E-13d

Appendix F

Appendix F

Performance Plots for the Extended Kalman Filter with Bias Correction Term

Constant Parameters

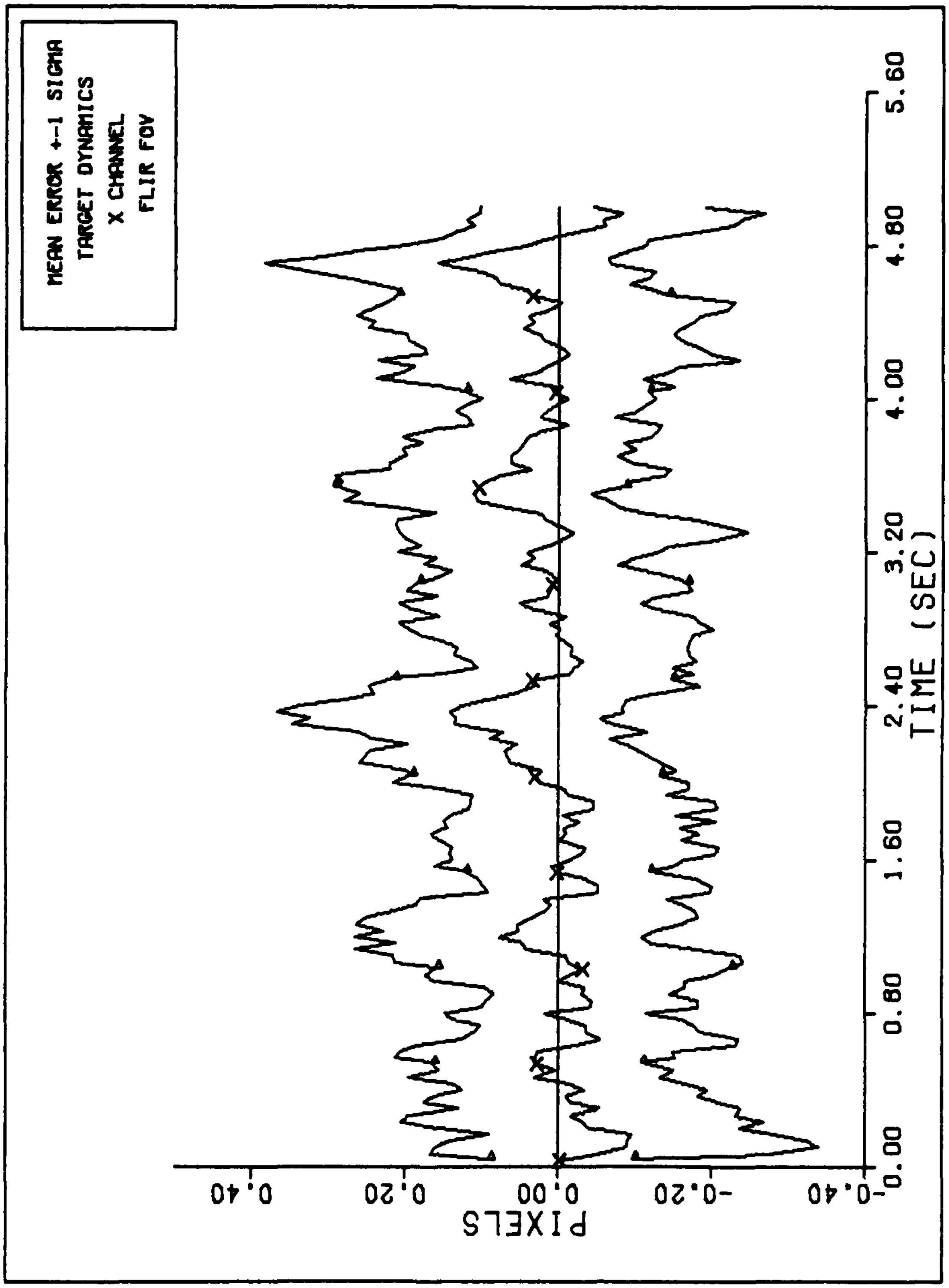
Symbol	FORTRAN Code	Valve
σ_D	SIGSI	0
σ_f	SIGFLR	0
σ_A	SIGAT	0.2
$\sigma_{g_2 T}$	SIGMS	1
AR_T	ASPRO	5
	ISPTL	'NO'
	NRUN	20
	TFINAL	5
$\sigma_{\dot{X}}$	SIGMF0	-3
AR_F	AR0	1
σ_{AF}	SIGF2	0.2
σ_V	RF	2

Input Parameters

Symbol	Case Fortran Code	F1	F2	F3	F4	F5	F6	F7
I_{max}	IMAX	25	25	25	25	25	25	25
σ_N	SIGMAB	2	2	2	2	2	2	2
$I_{max}F$	FIMAX0	25	25	25	25	25	25	25
σ_{DF}	SIGFI0	500	500	1000	5000	1000	1000	10,000
Acquisition $\underline{Q}_{FD}$		NO	NO	NO	NO	NO	NO	YES
Turn SIGF1		-	-	-	-	20,000	50,000	-
S/N	SN	12.5	12.5	12.5	12.5	12.5	12.5	12.5
Turn (G)		0	5	10	10	20	20	0
Initial $x_d(0)$	Conditions XFP(1)	0	0	0	0	0	0	1.46
$y_d(0)$	SFP(2)	0	0	0	0	0	0	1.43
$\dot{x}_e$	XDOT	-530.33	-750	-750	-750	-750	-750	-530.33
$\dot{y}_e$	YDOT	-530.33	0	0	0	0	0	-530.33
$\dot{z}_e$	ZDOT	0	0	0	0	0	0	-530.33

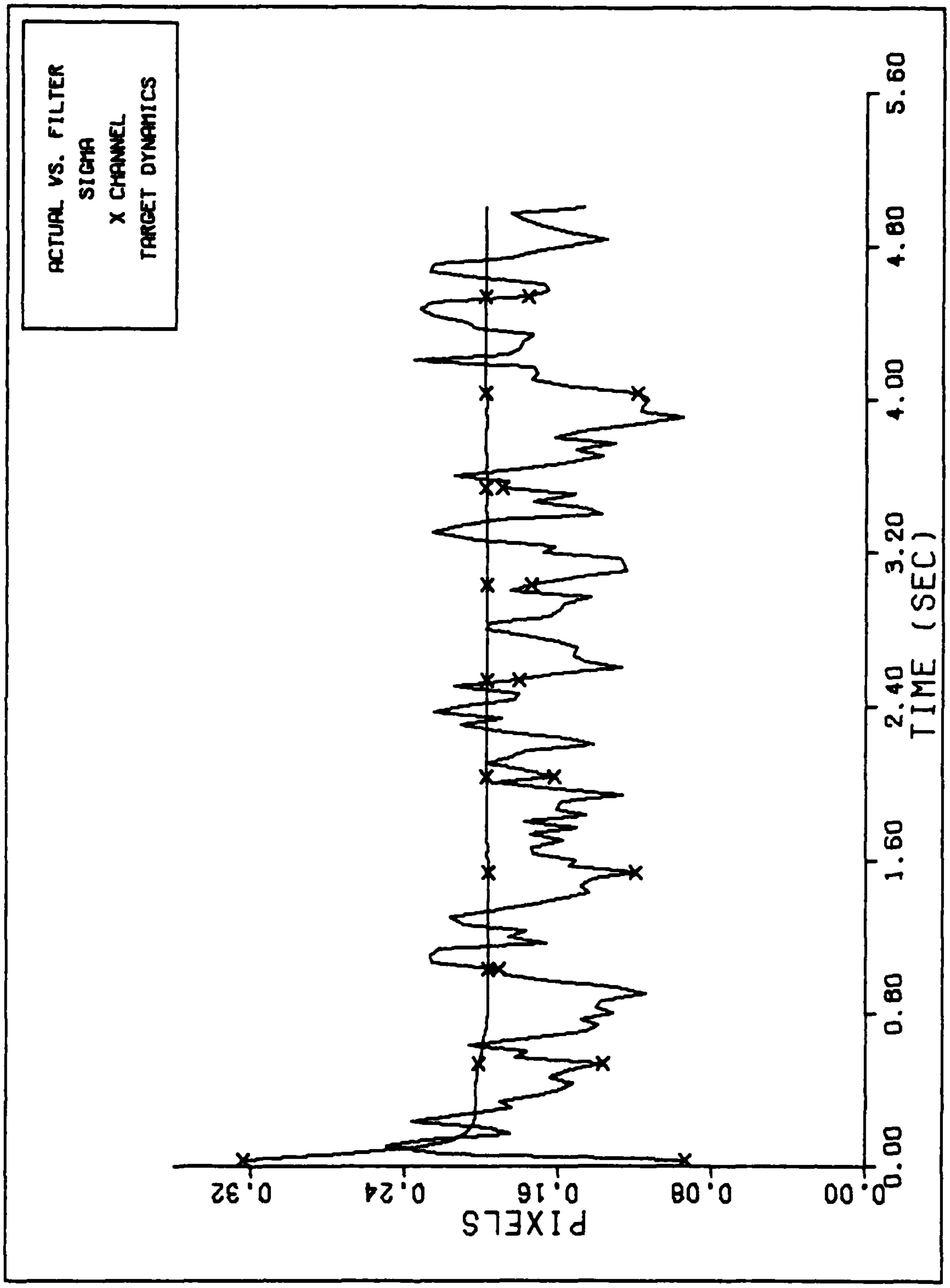
Input Parameters

Symbol	Case / Fortran Code	F8	F9	F10	F11	F12	F13
I_{max}	IMAX	25	25	10	4	25	10
σ_N	SIGMAB	2	2	2	2	12.5	5
$I_{max}F$	FIMAX0	25	25	10	4	25	20
σ_{DF}	SIGFI0	10,000	10,000	500	500	20,000	20,000
Acquisition Q_{FD}		YES	YES	NO	NO	NO	NO
Turn SIGF1		-	-	-	-	-	-
S/N	SN	12.5	12.5	5	2	2	2
Turn (G)		0	0	0	0	0	0
Initial Conditions $x_d(0)$	XFP(1)	0	0.5	0	0	0	0
$y_d(0)$	SFP(2)	0	0.5	0	0	0	0
$\dot{x}_e$	XDOT	-517.55	-525	-530.33	-530.33	-530.33	-530.33
$\dot{y}_e$	YDOT	-543.11	-535	-530.33	-530.33	-530.33	-530.33
$\dot{z}_e$	ZDOT	5	8	0	0	0	0



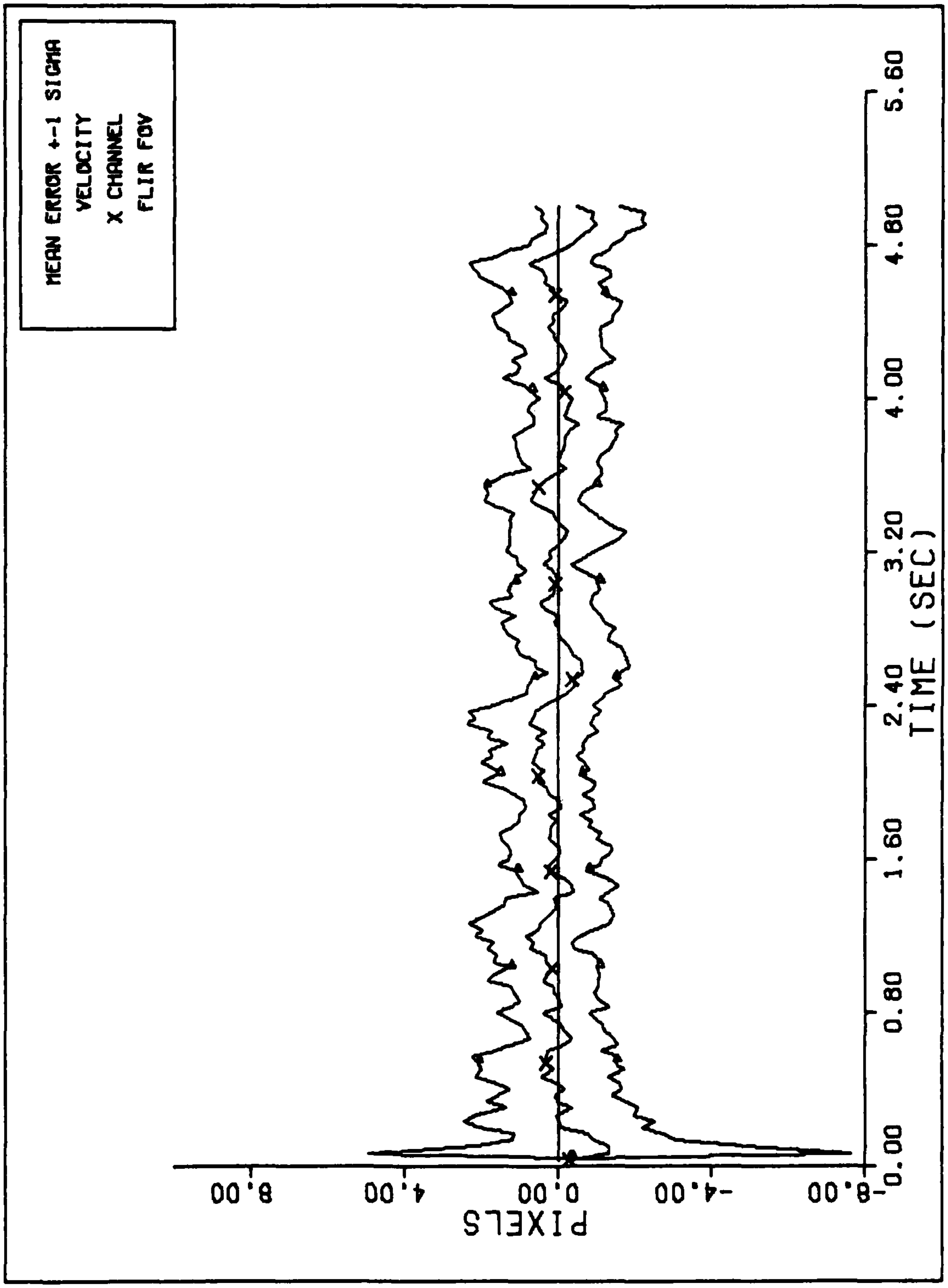
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure F-1a



FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

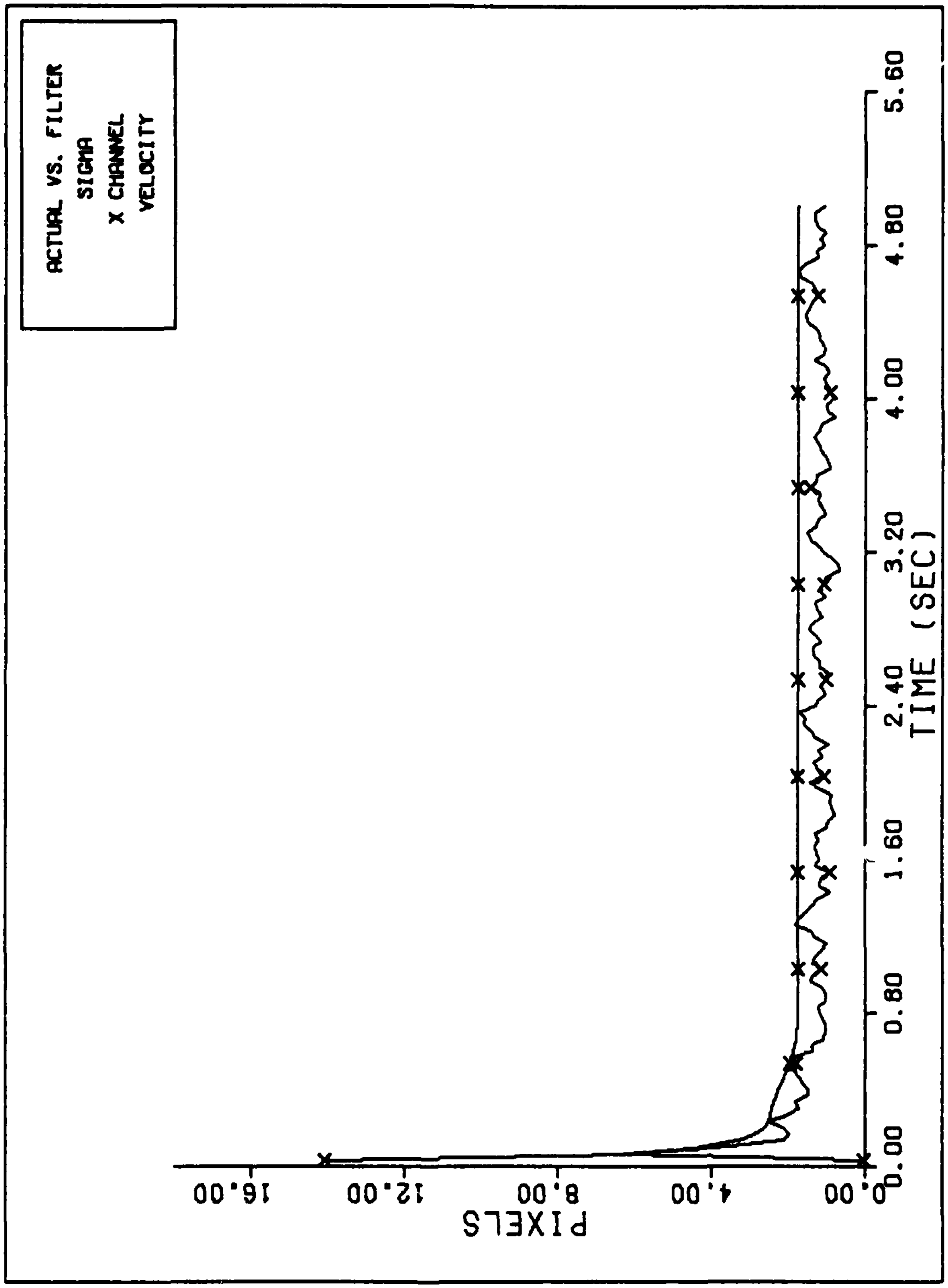
Figure F-1b



X CHANNEL VELOCITY ERROR (S/N=12.5)

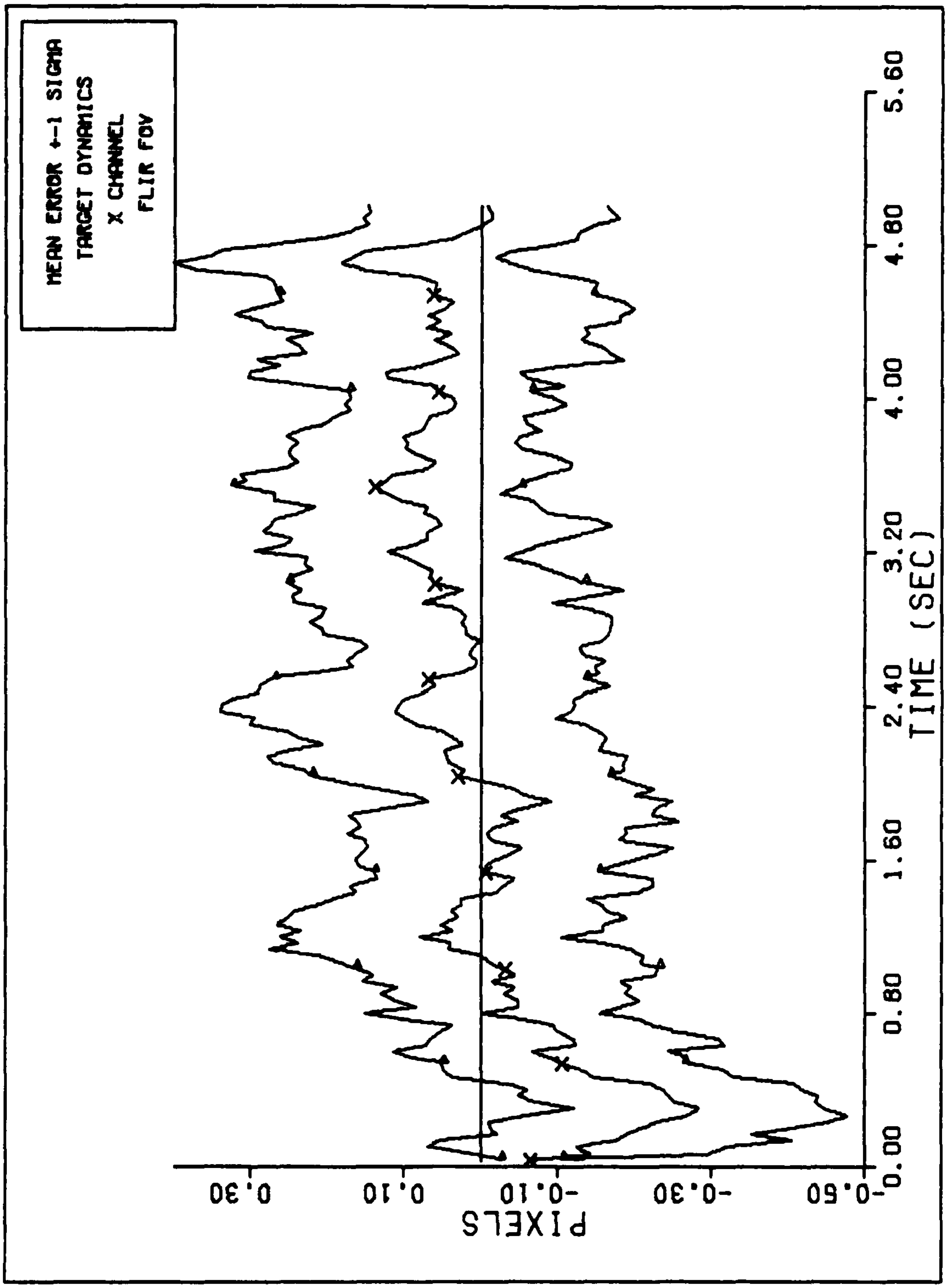
Figure F-1c

F1c



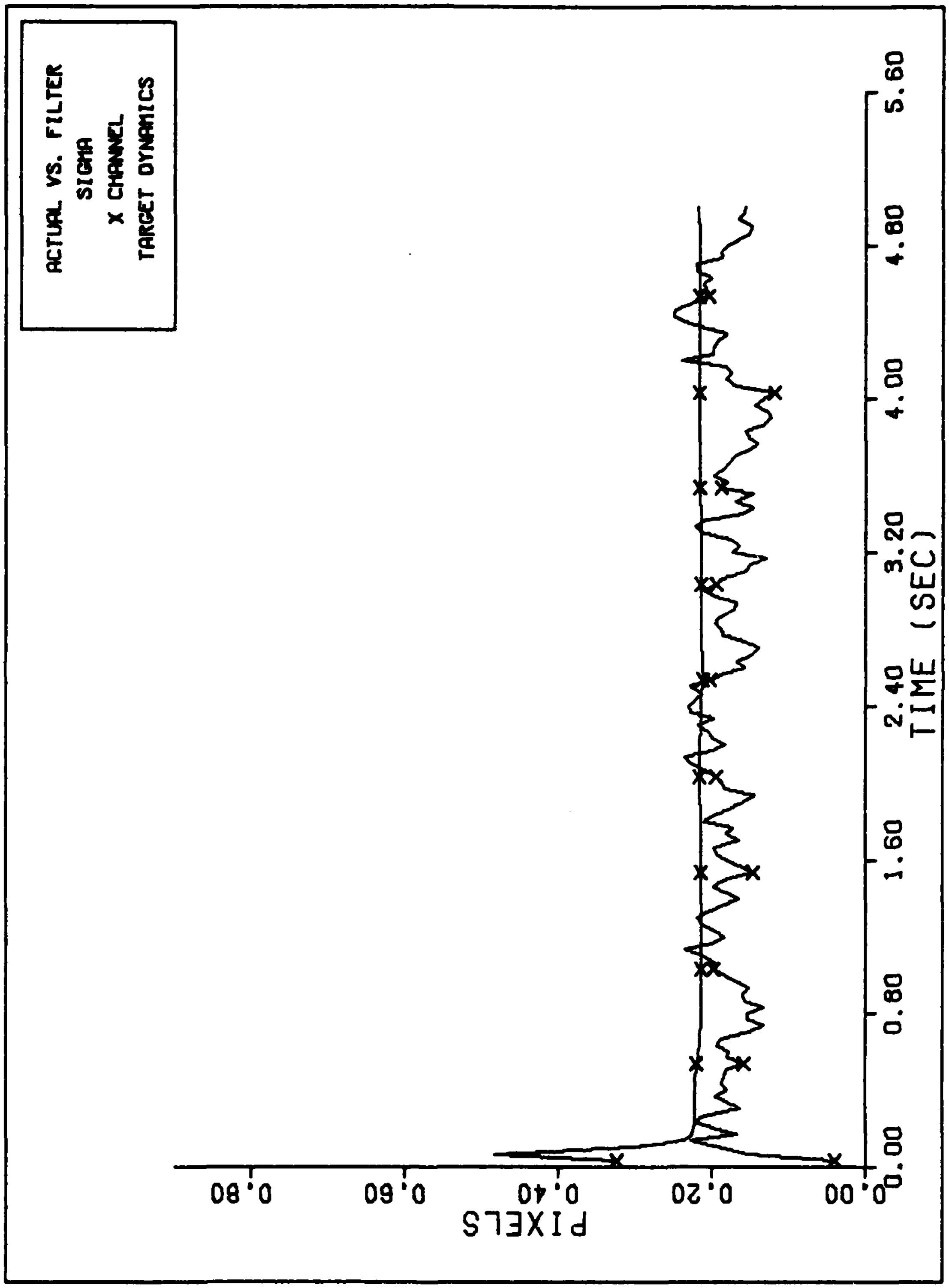
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-1d



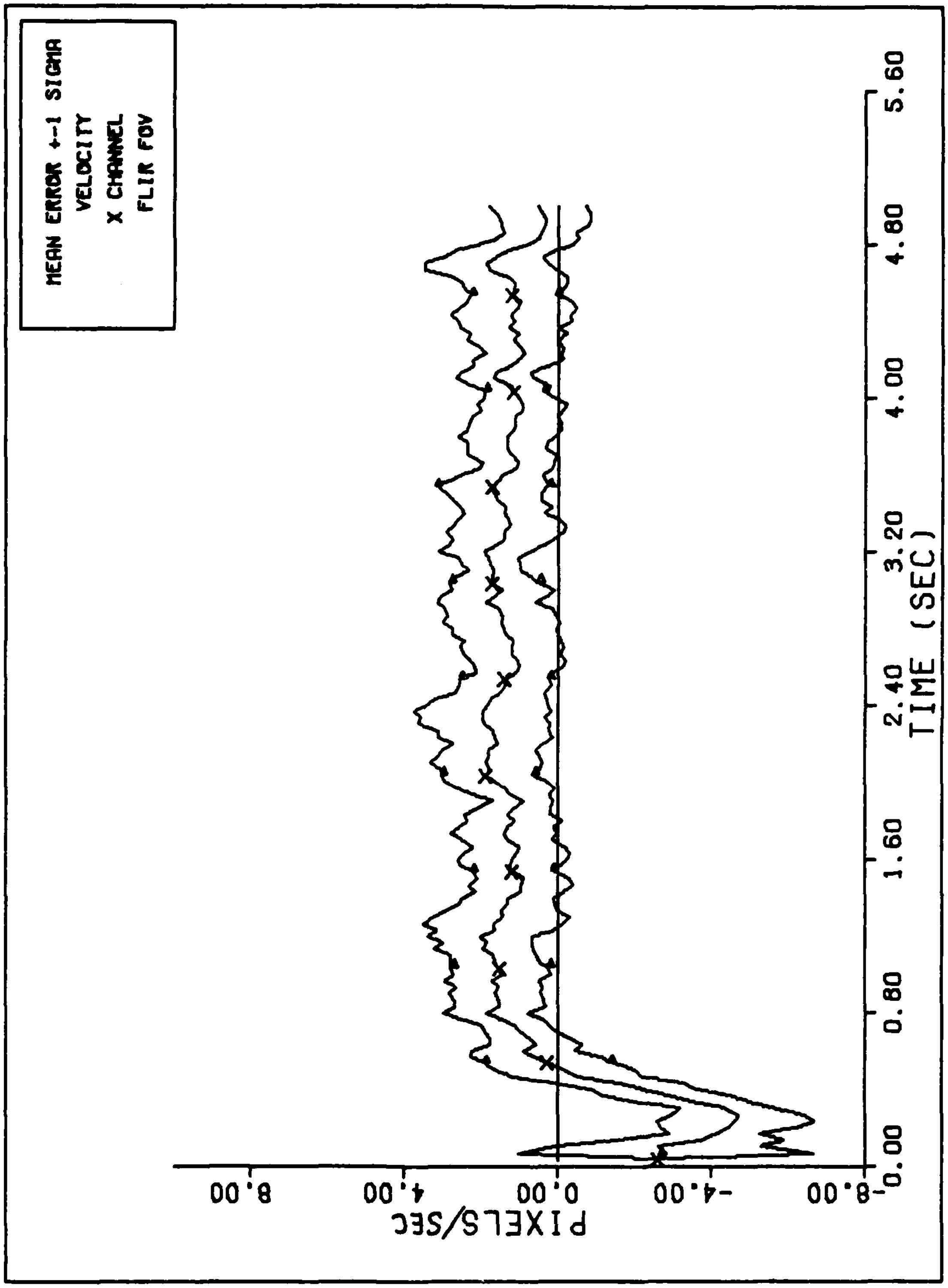
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure F-2a



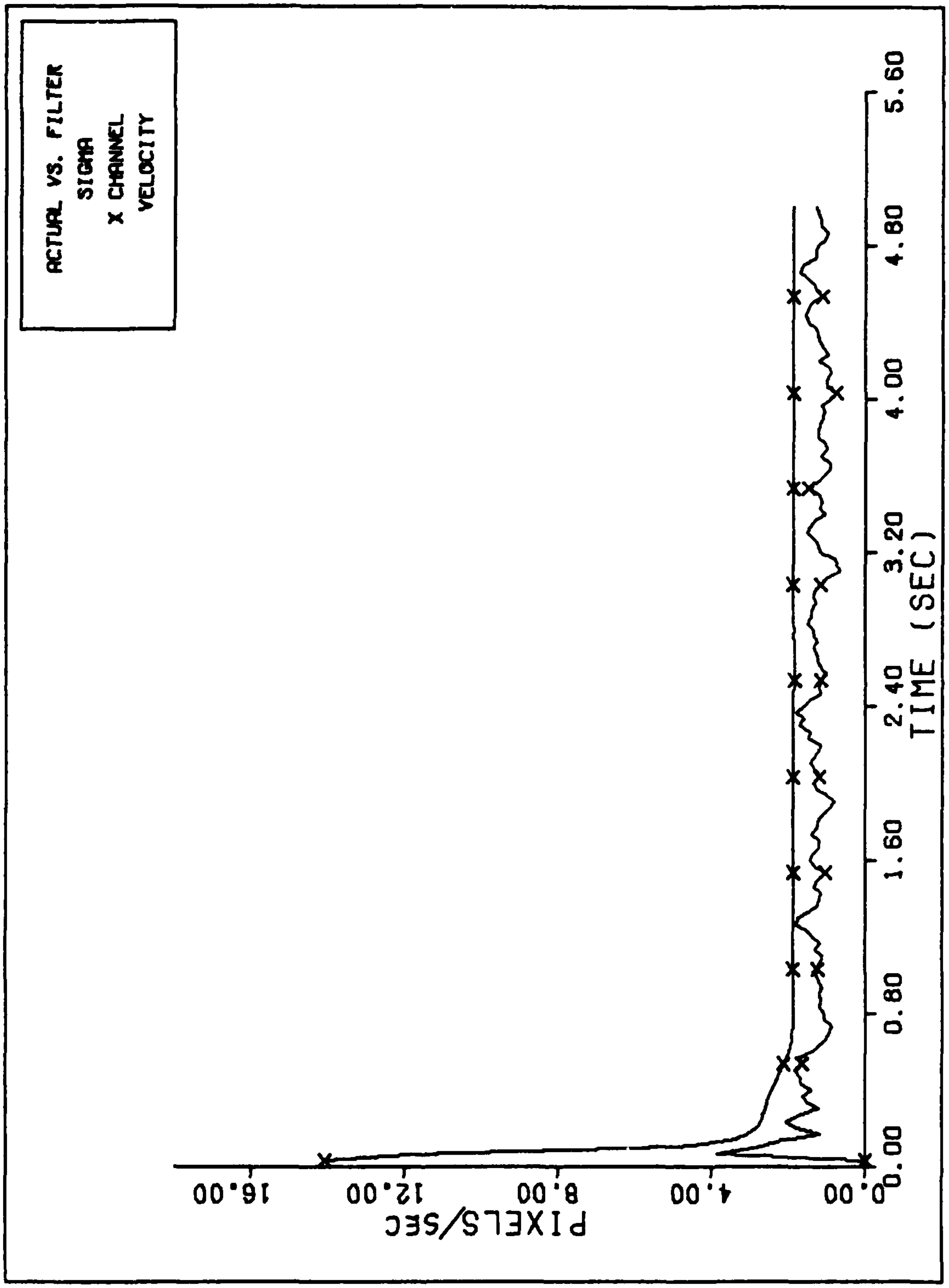
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-2b



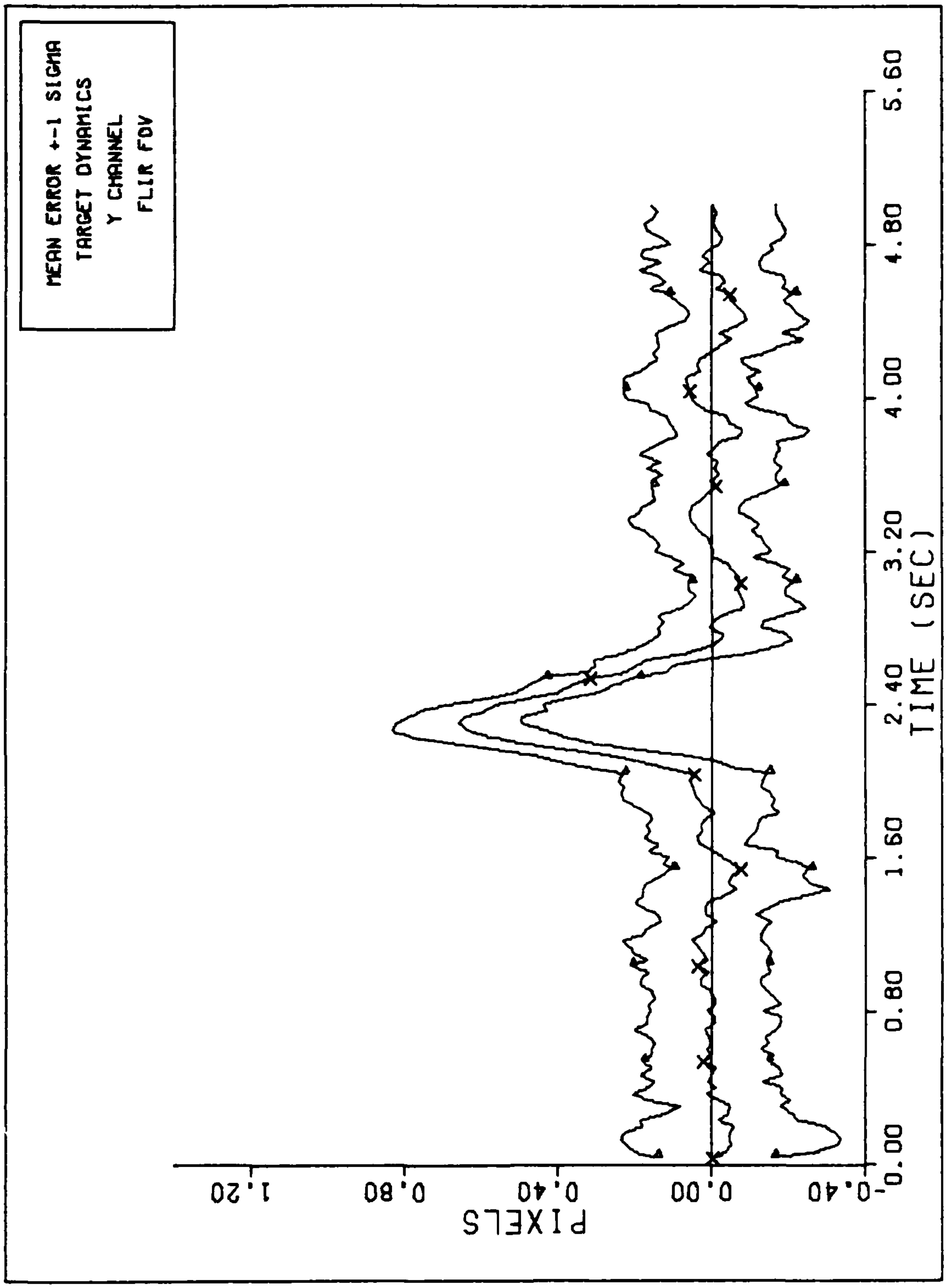
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure F-2c



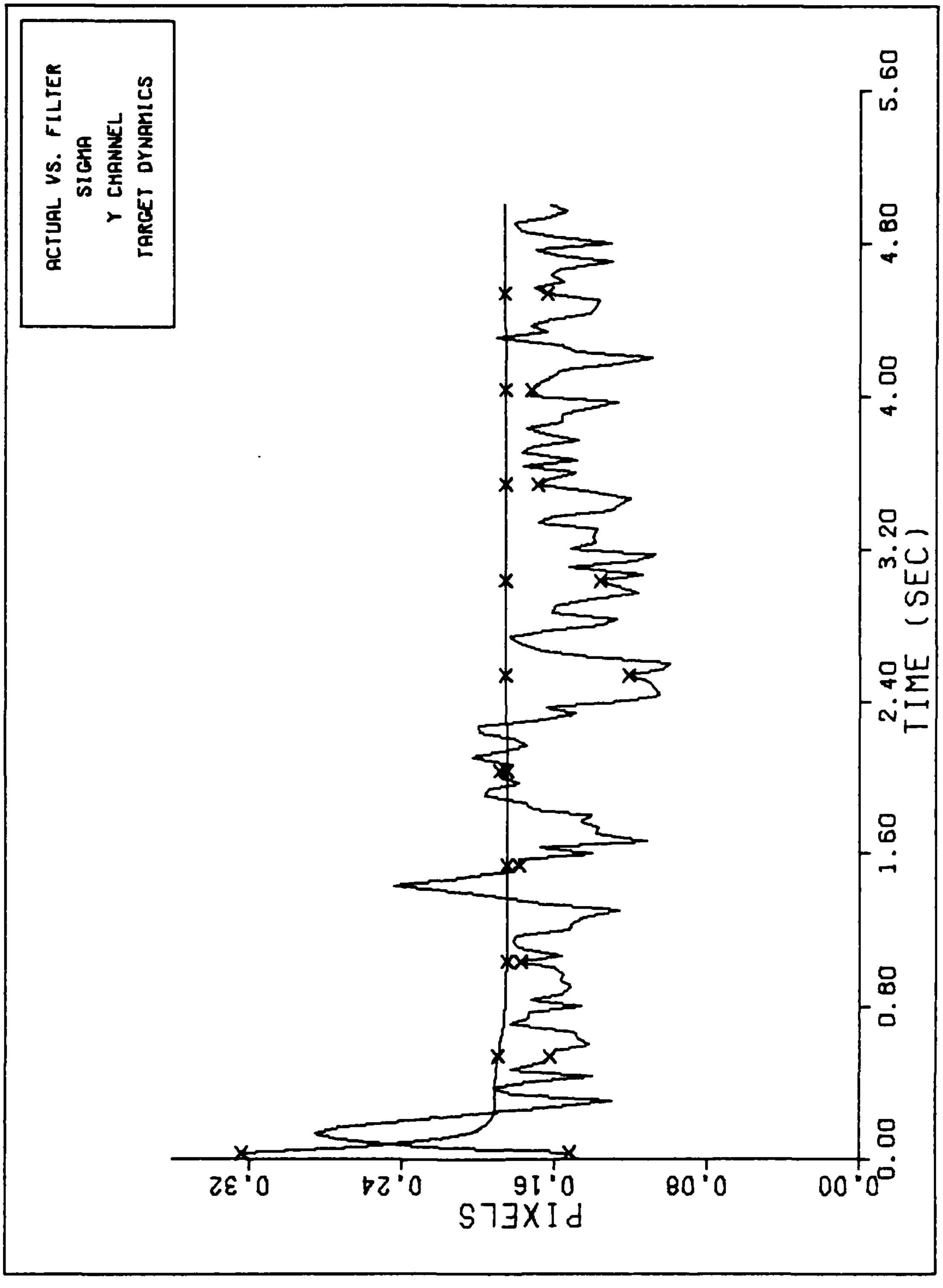
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-2d



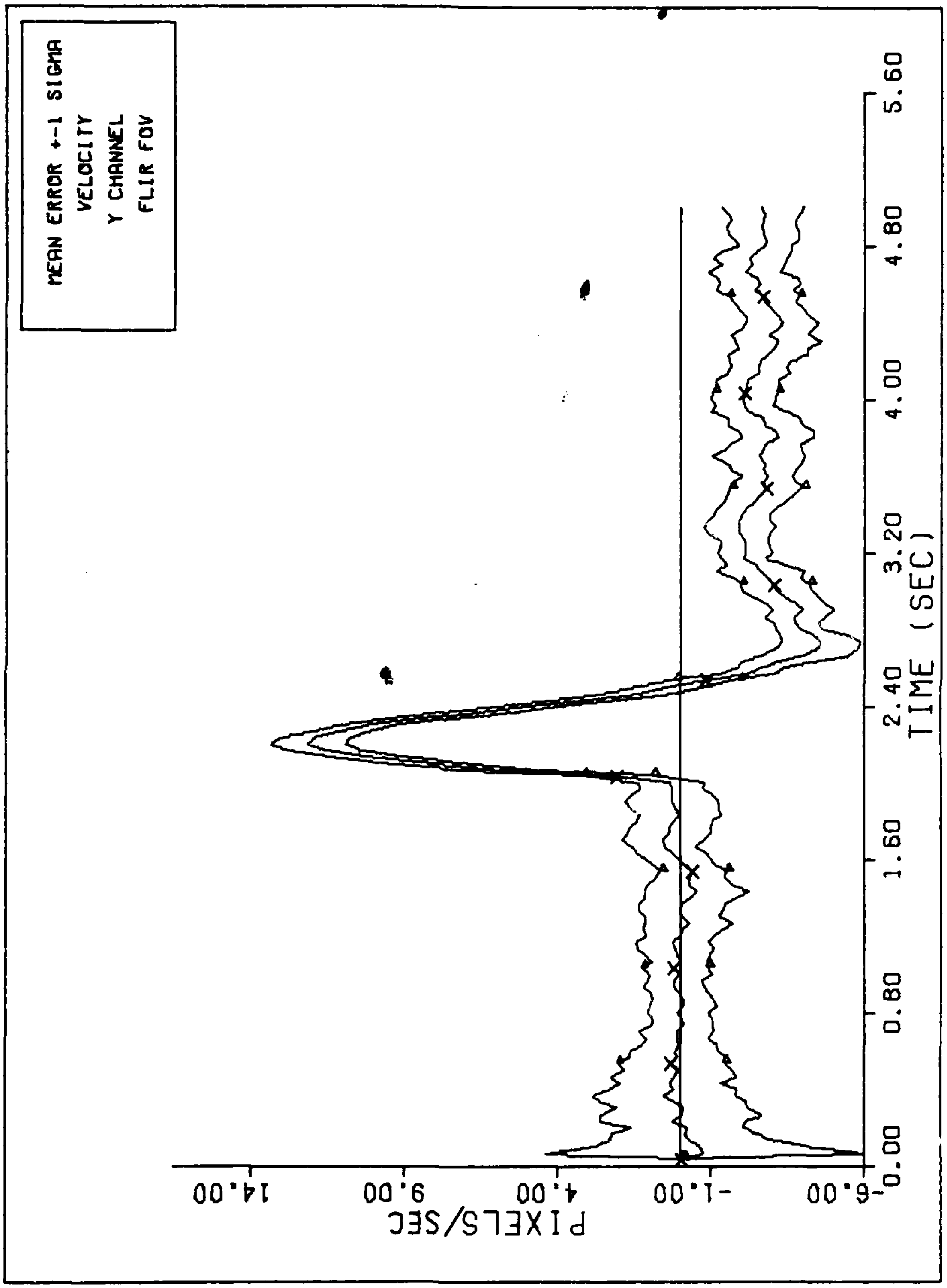
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure F-2e



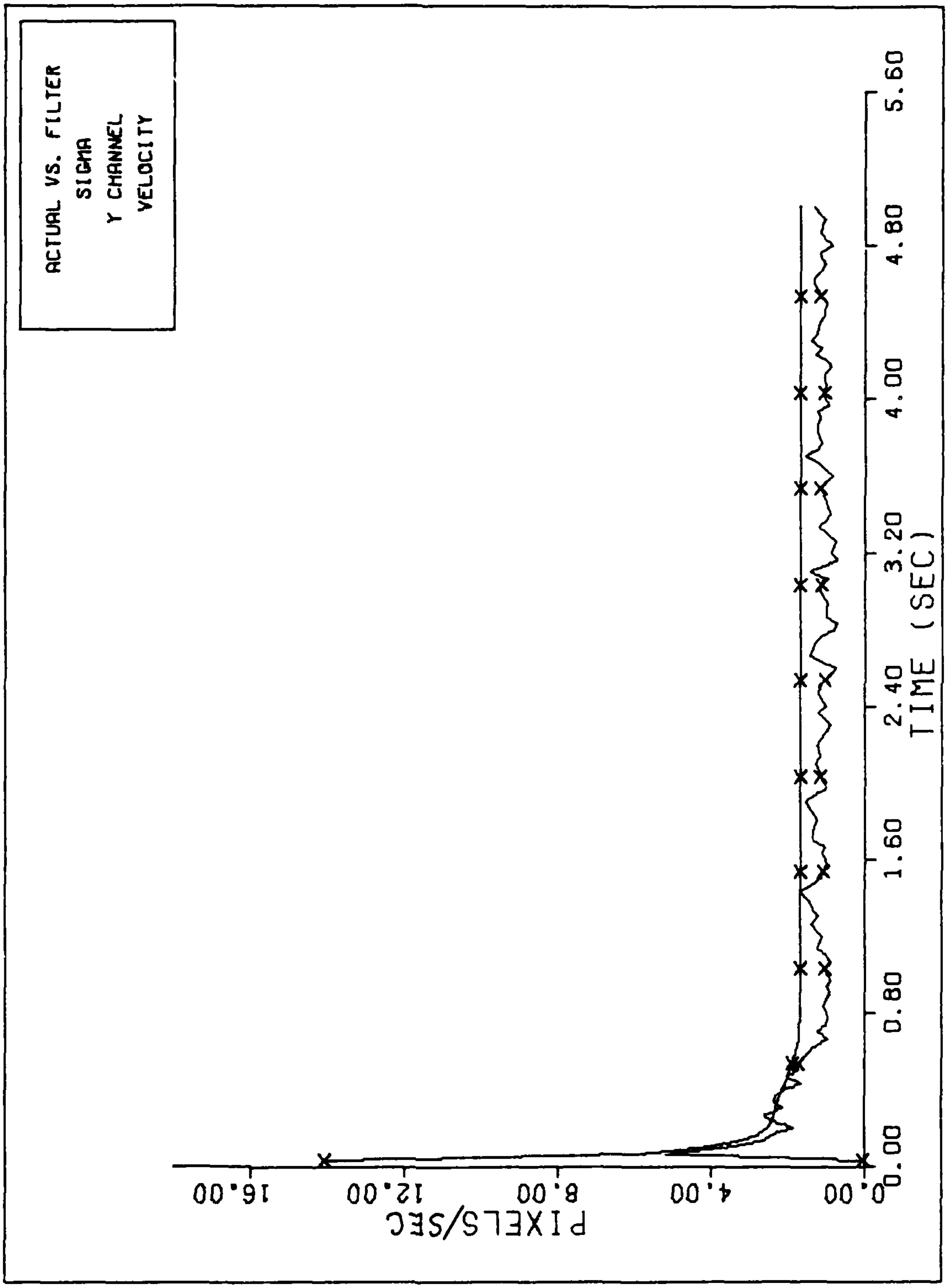
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-2f



Y CHANNEL VELOCITY ERROR (S/N=12.5)

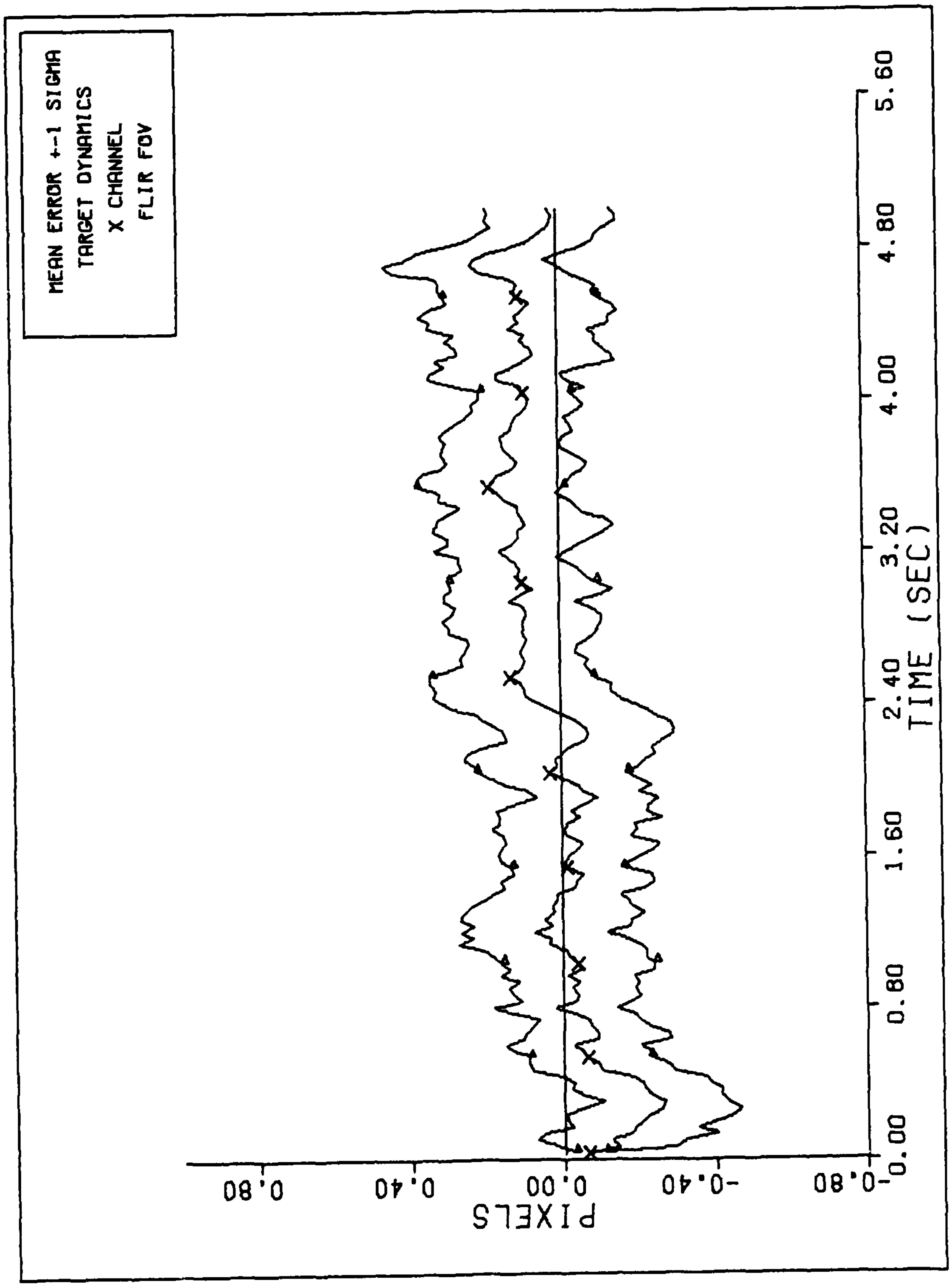
Figure F-2g



FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

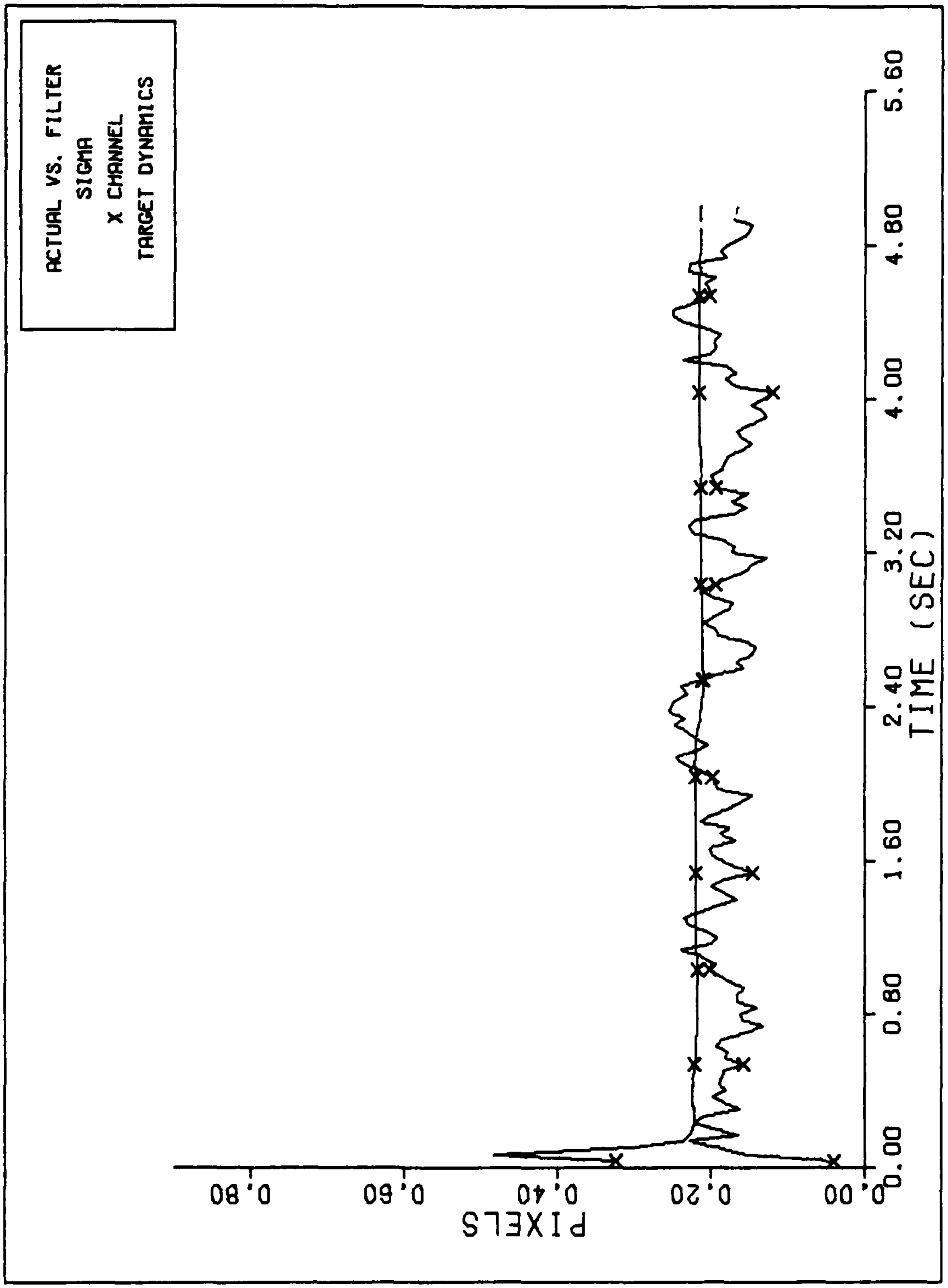
Figure F-2h

F2h



X CHANNEL DYNAMICS ERROR (S/N=12.5)

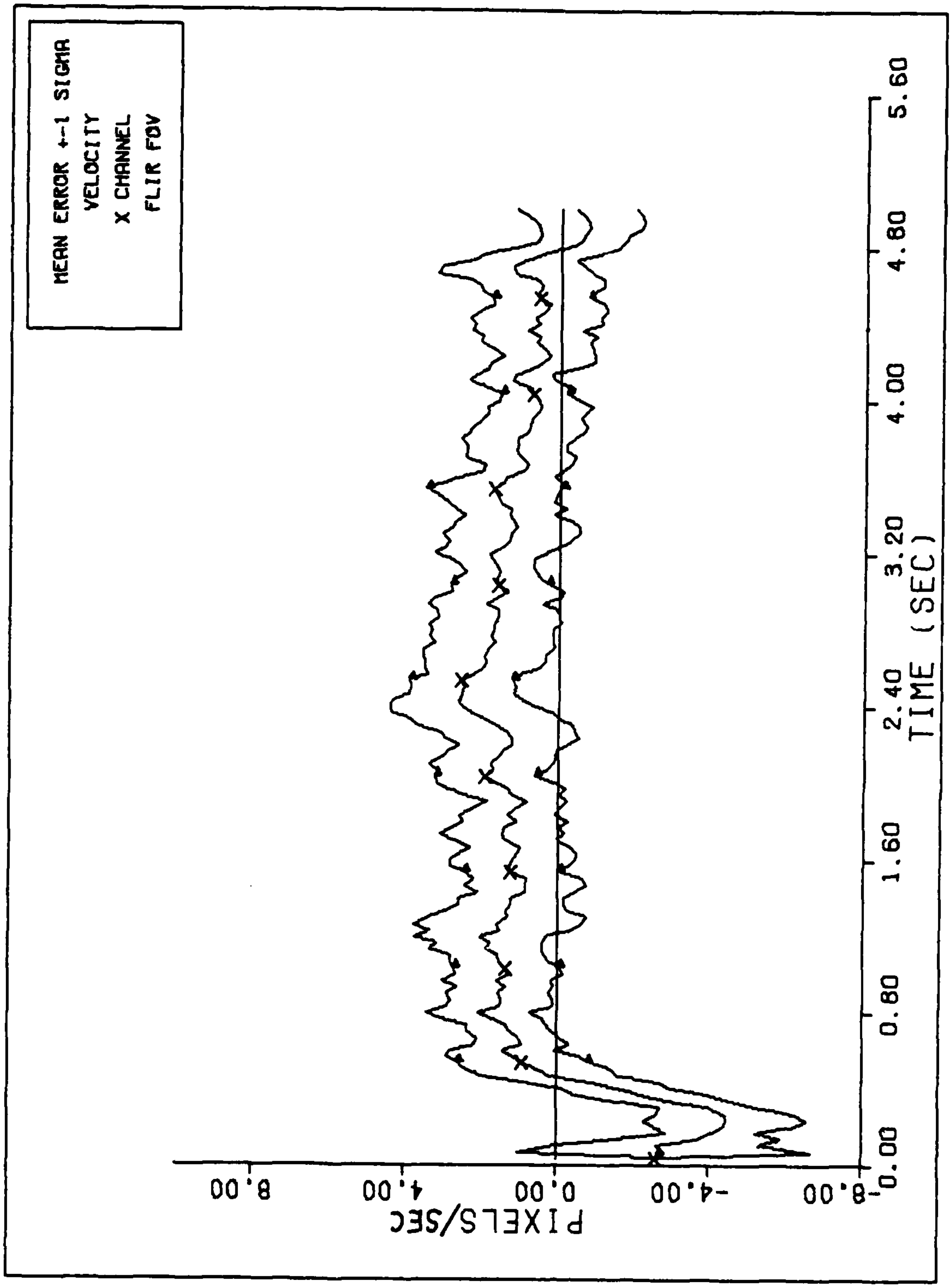
Figure F-3a



FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

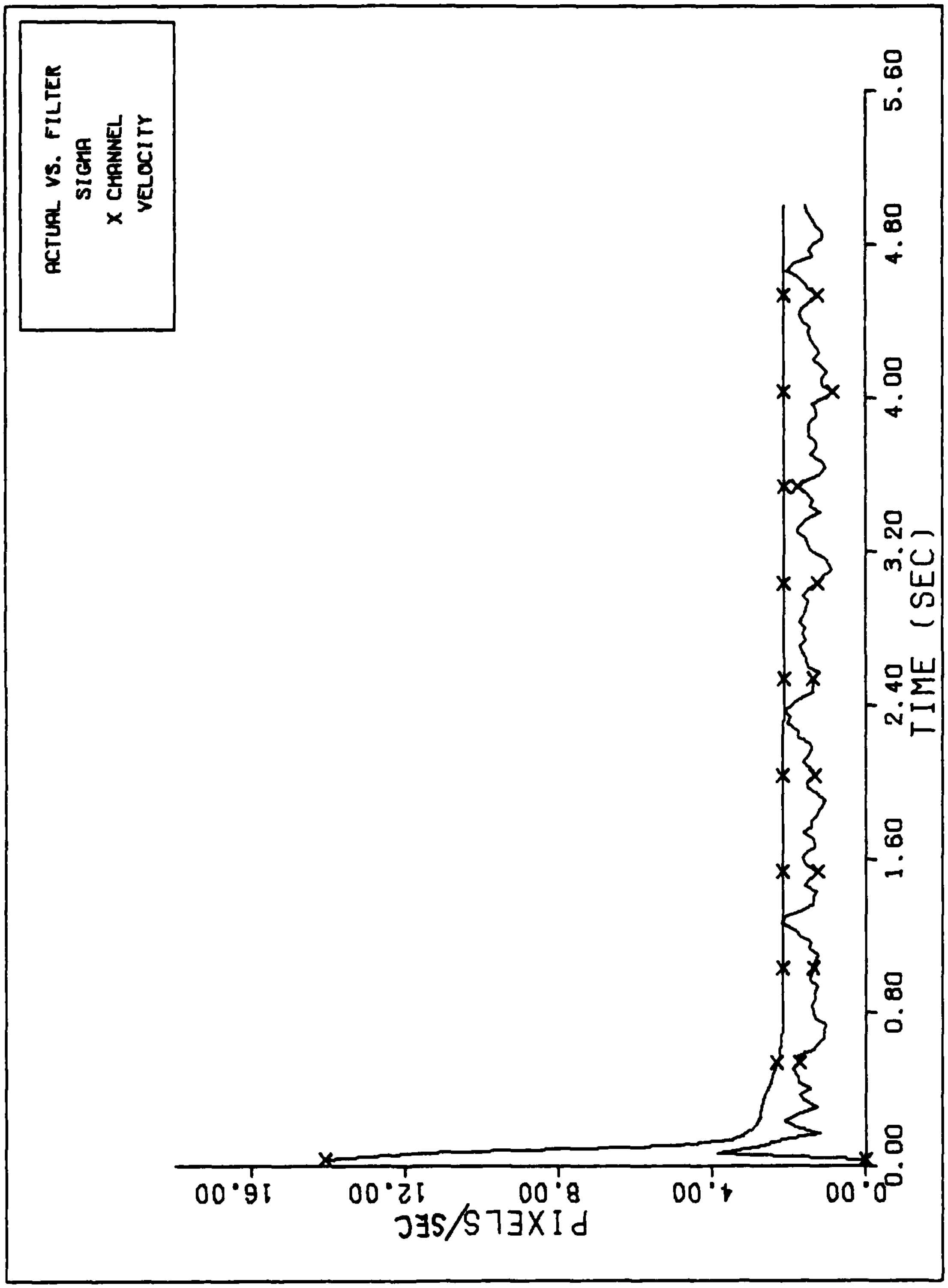
Figure F-3b

F3b



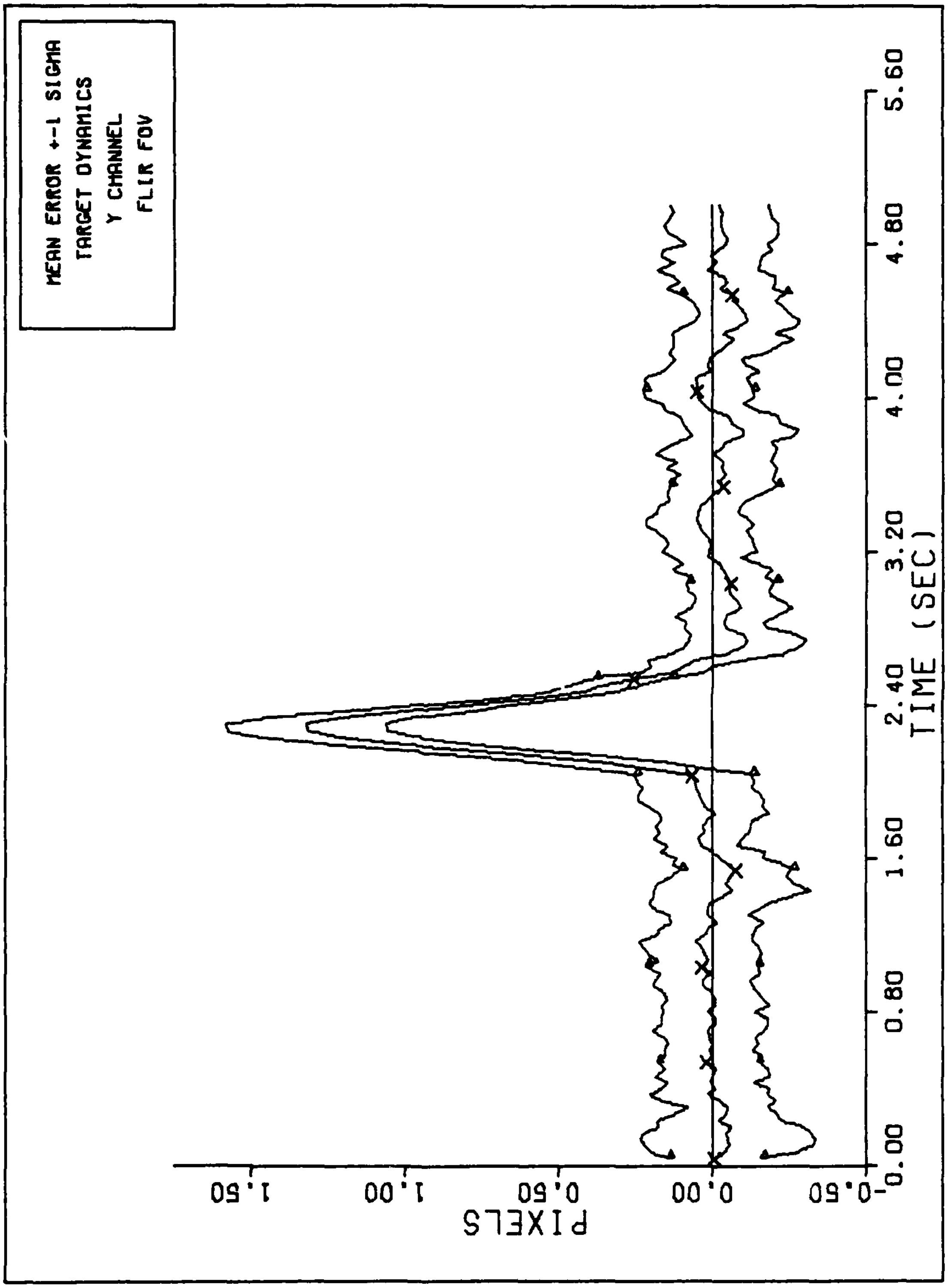
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure F-3c



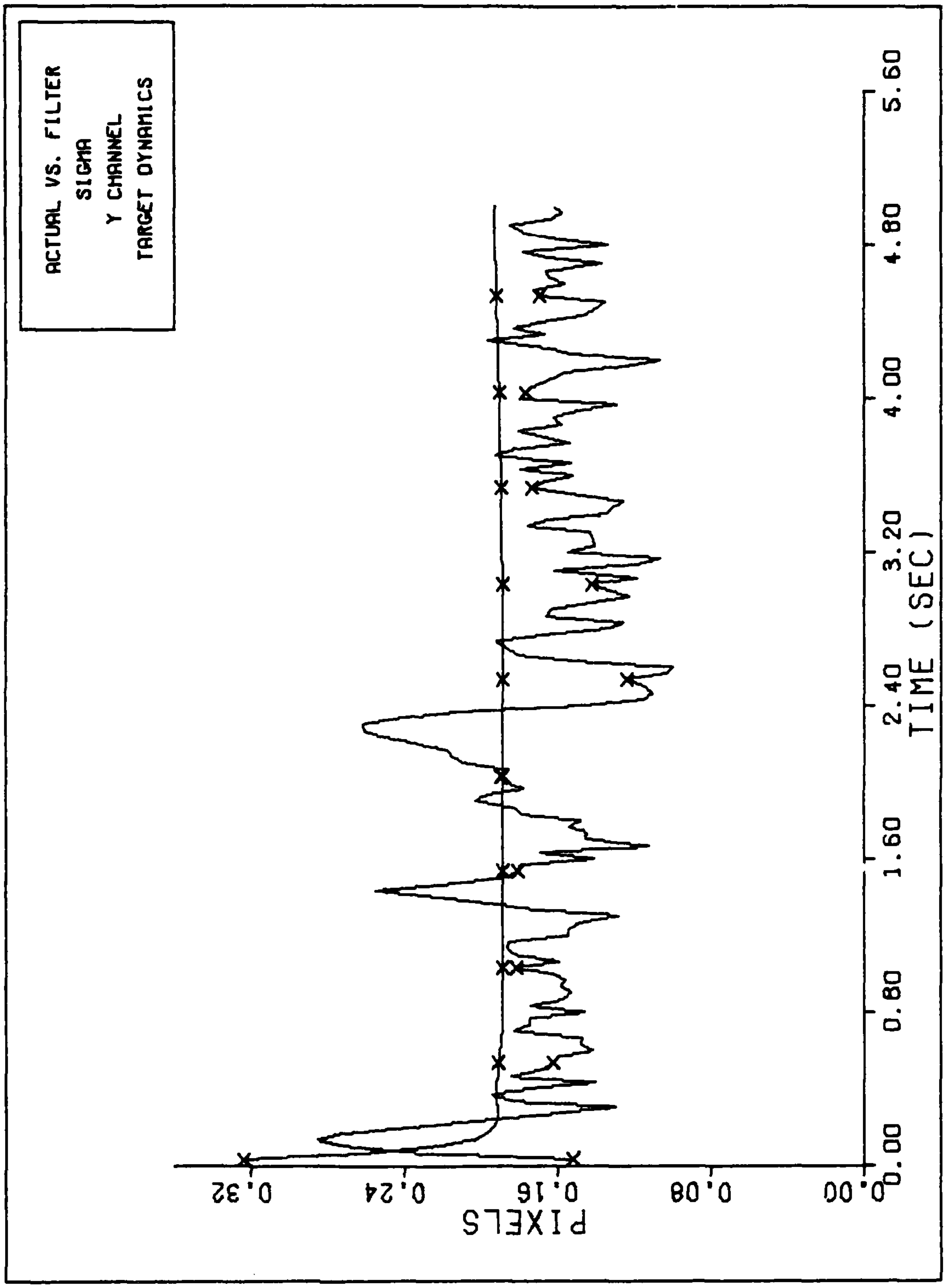
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-3d



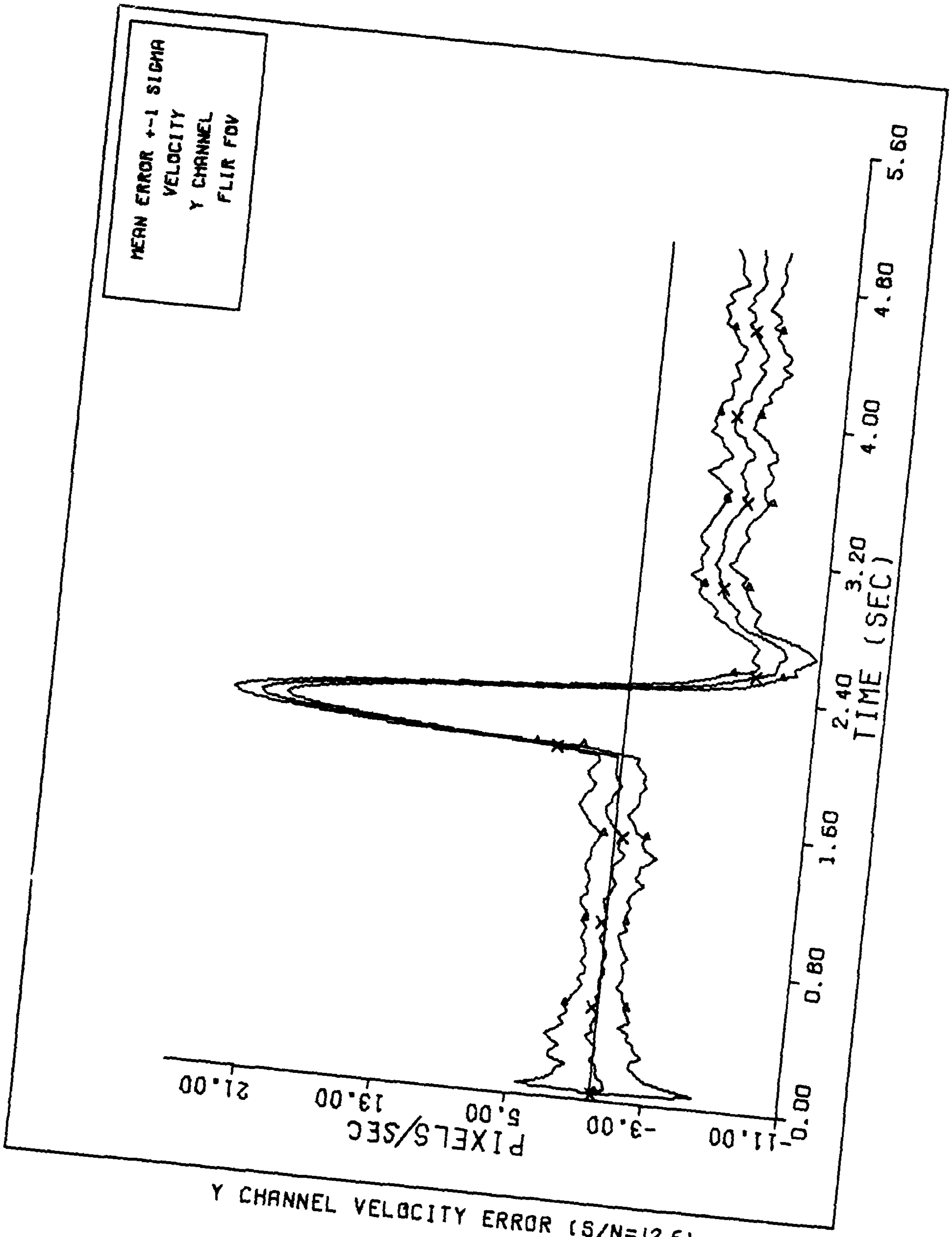
Y CHANNEL DYNAMICS ERROR (S/N=125)

Figure F-3e



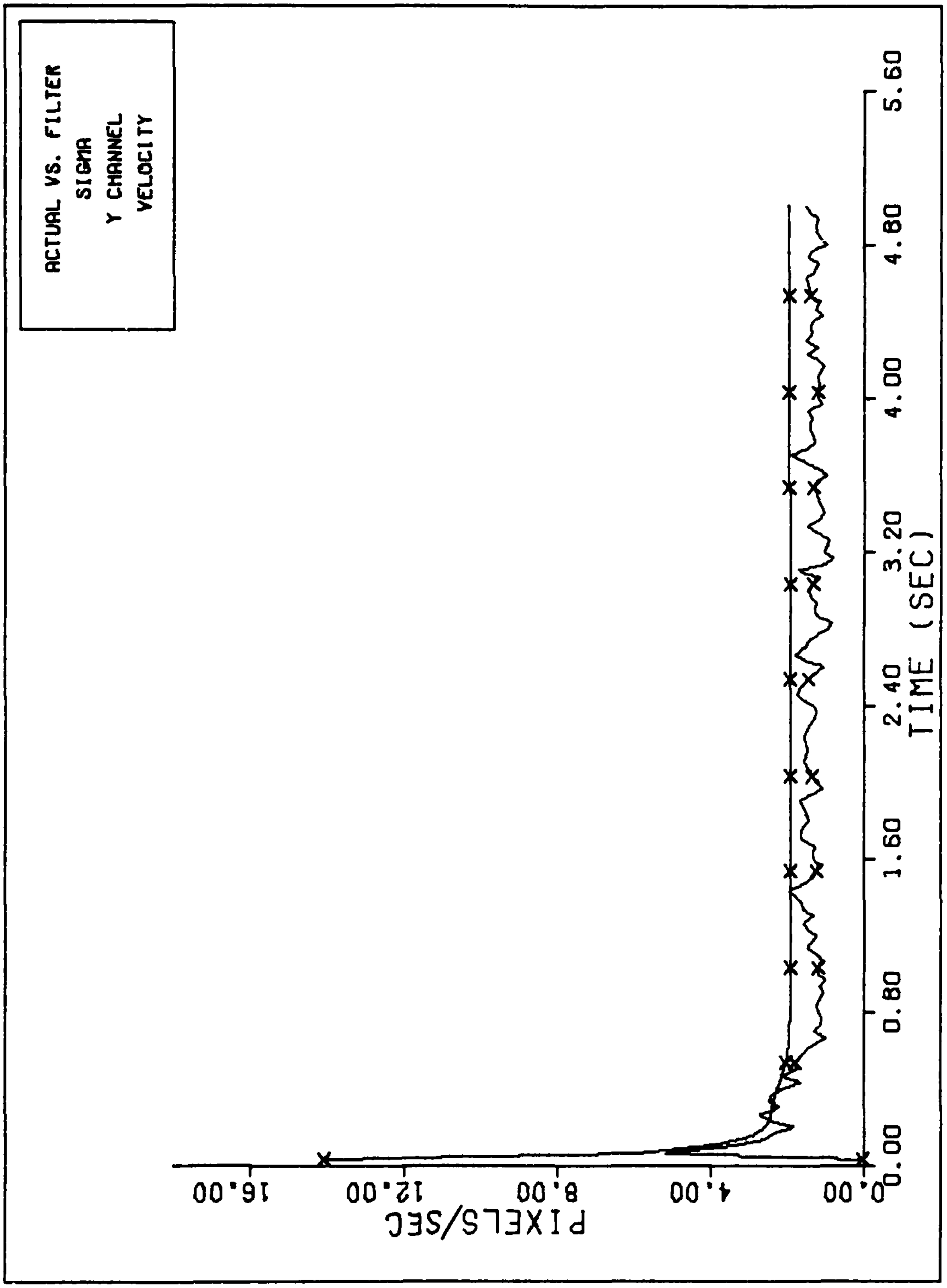
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-3f



Y CHANNEL VELOCITY ERROR (S/N=12.5)

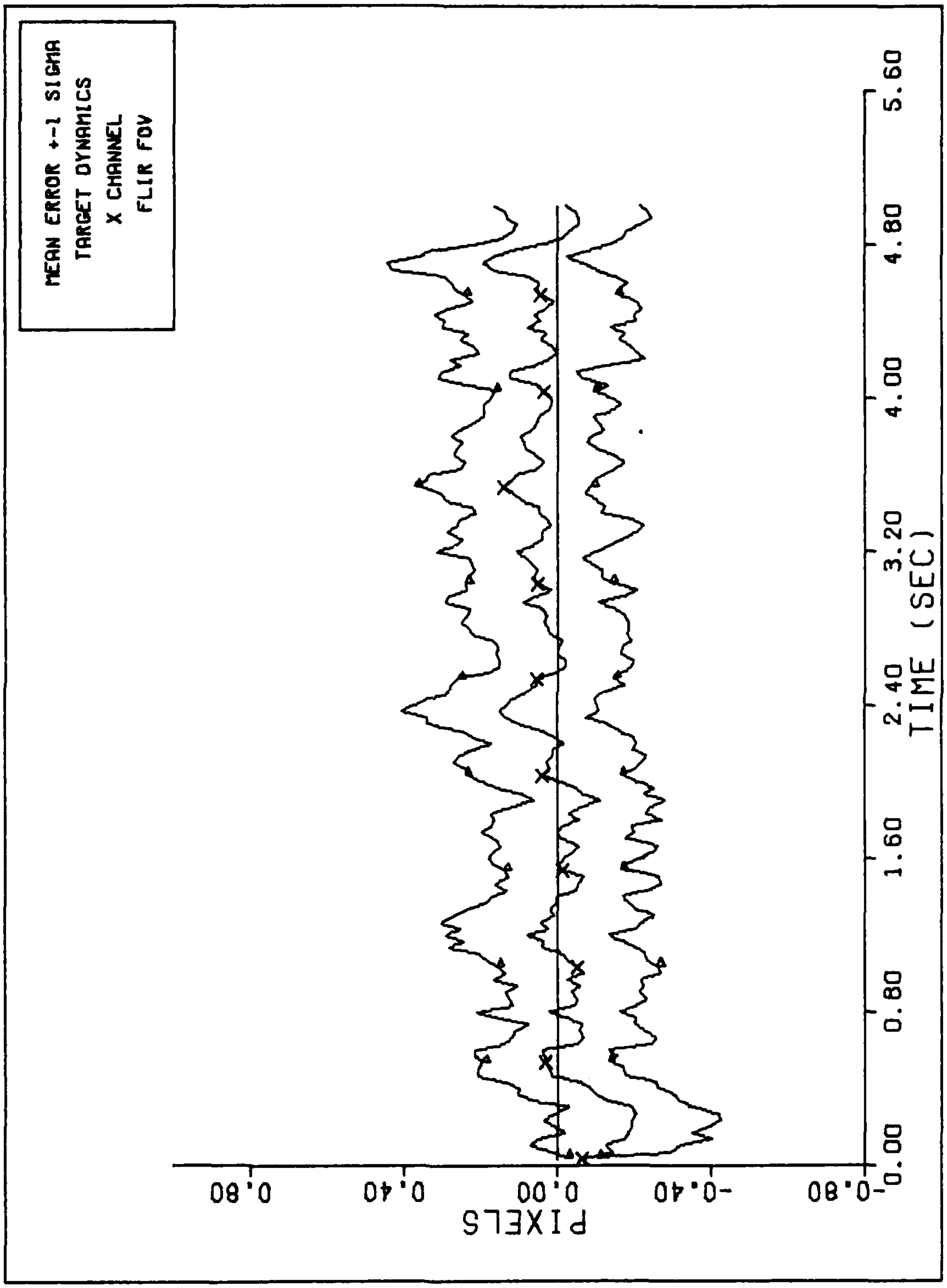
Figure F-3g



FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-3h

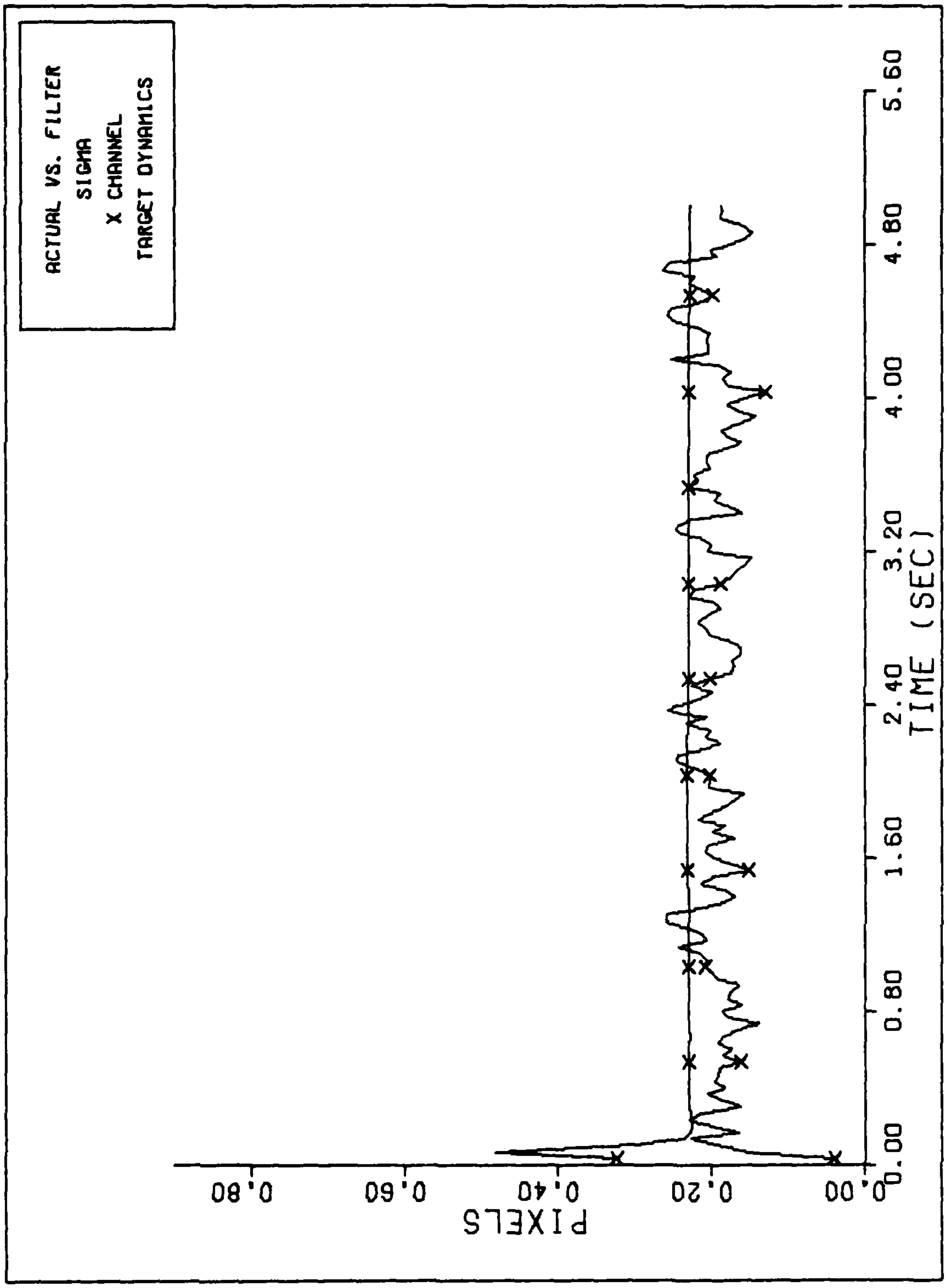
F3h



X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure F-4a

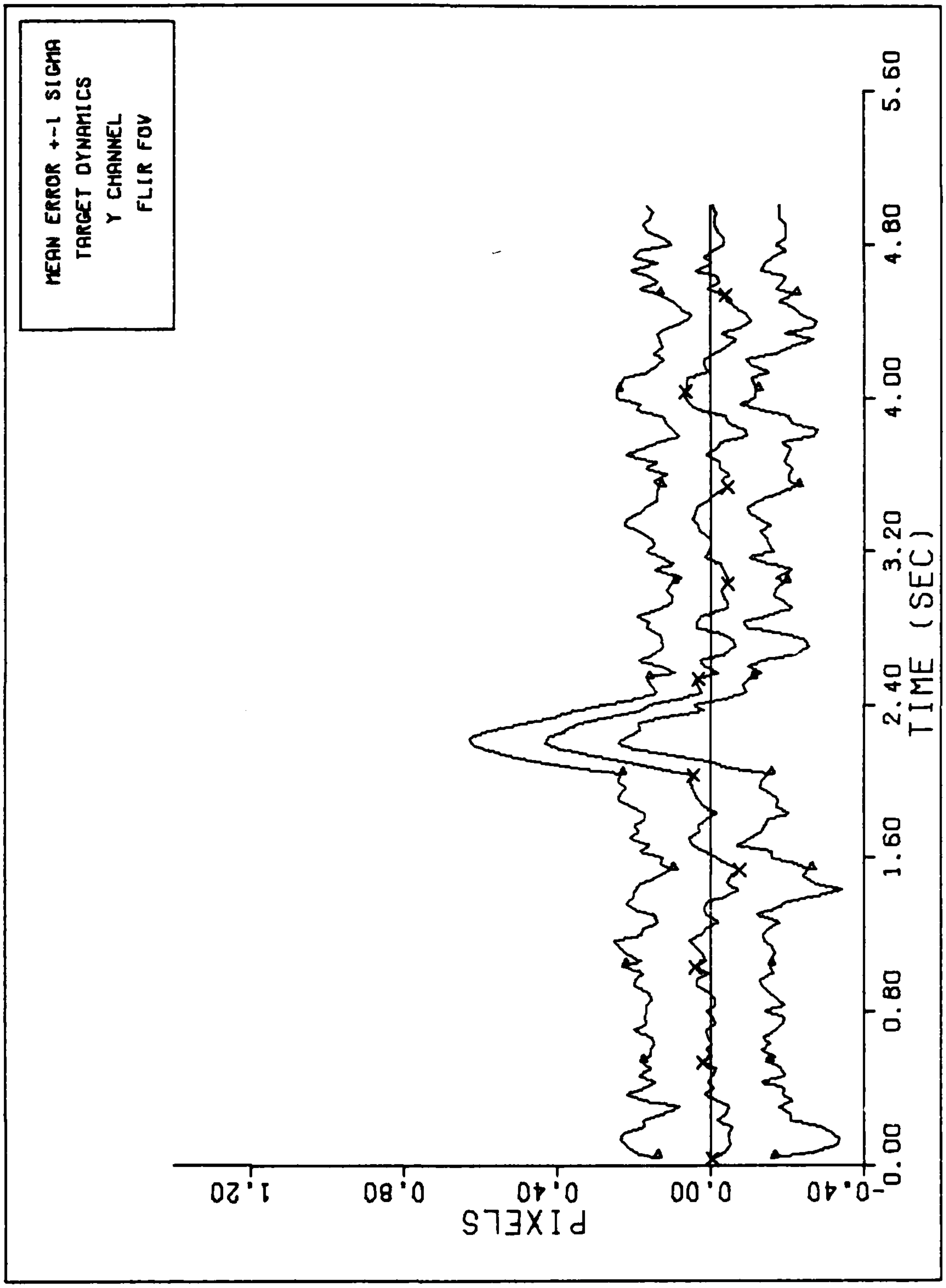
F4a



FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-4b

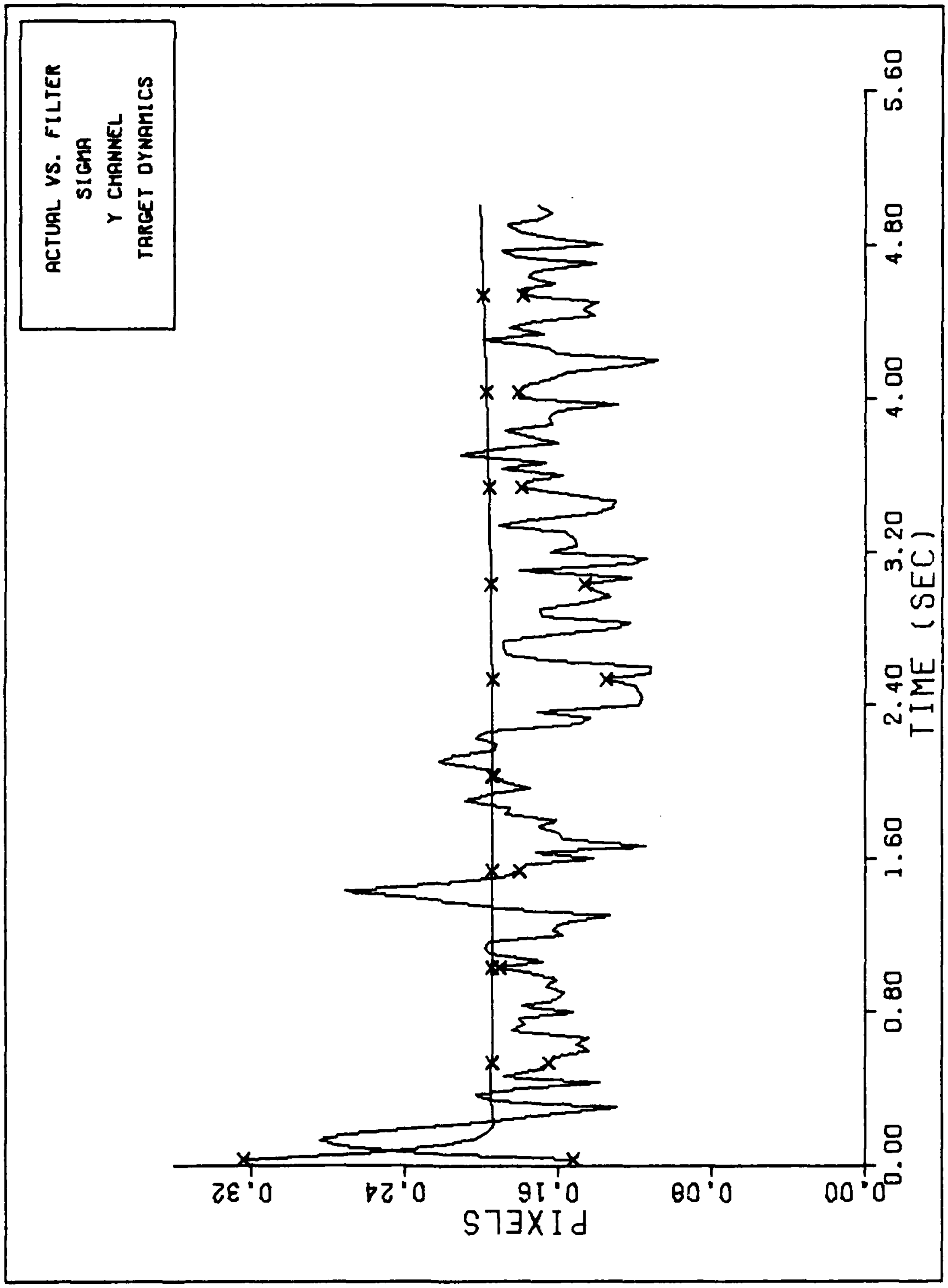
F4b



Y CHANNEL DYNAMICS ERROR (S/N=12.5)

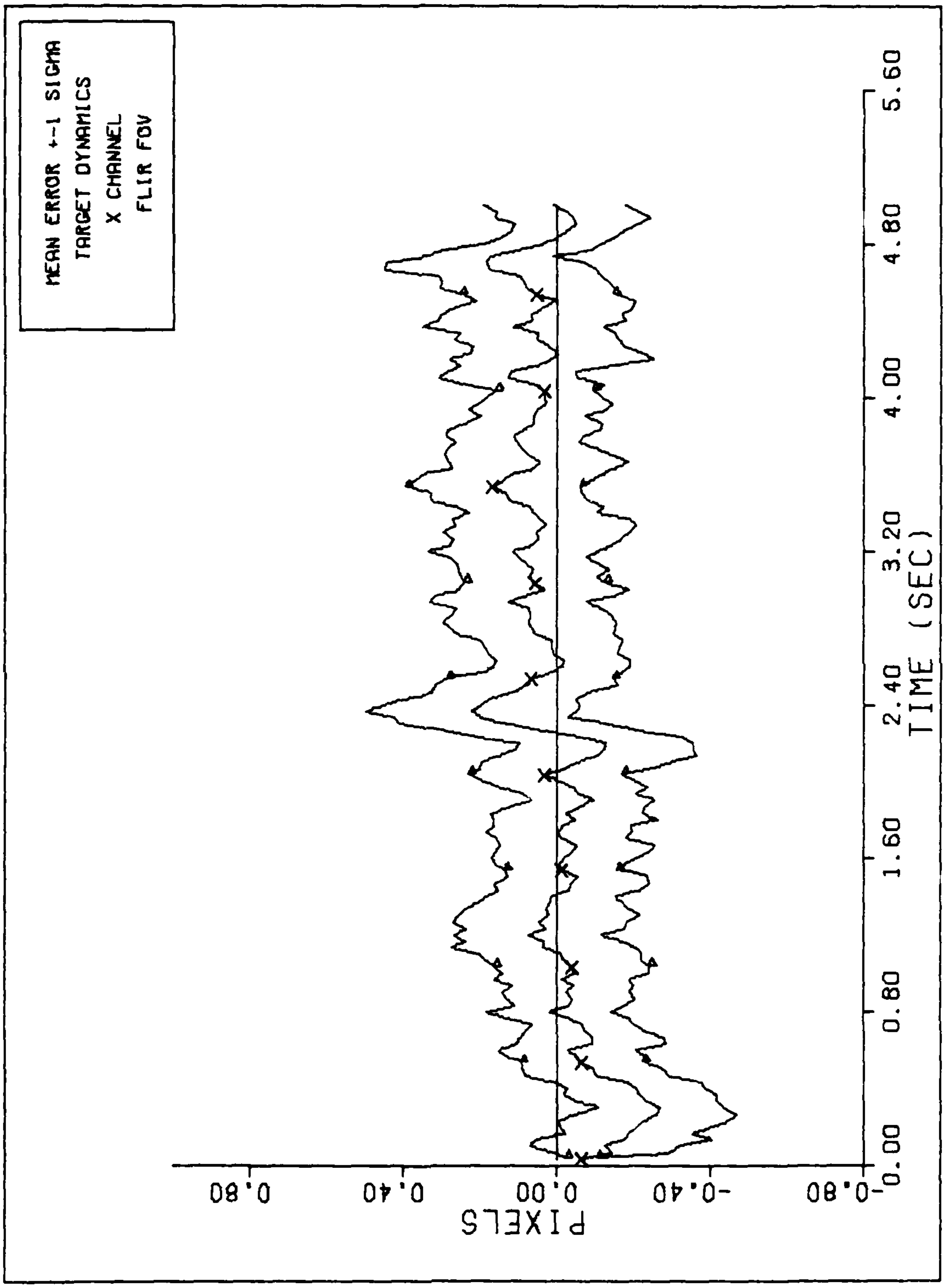
Figure F-4c

F4c



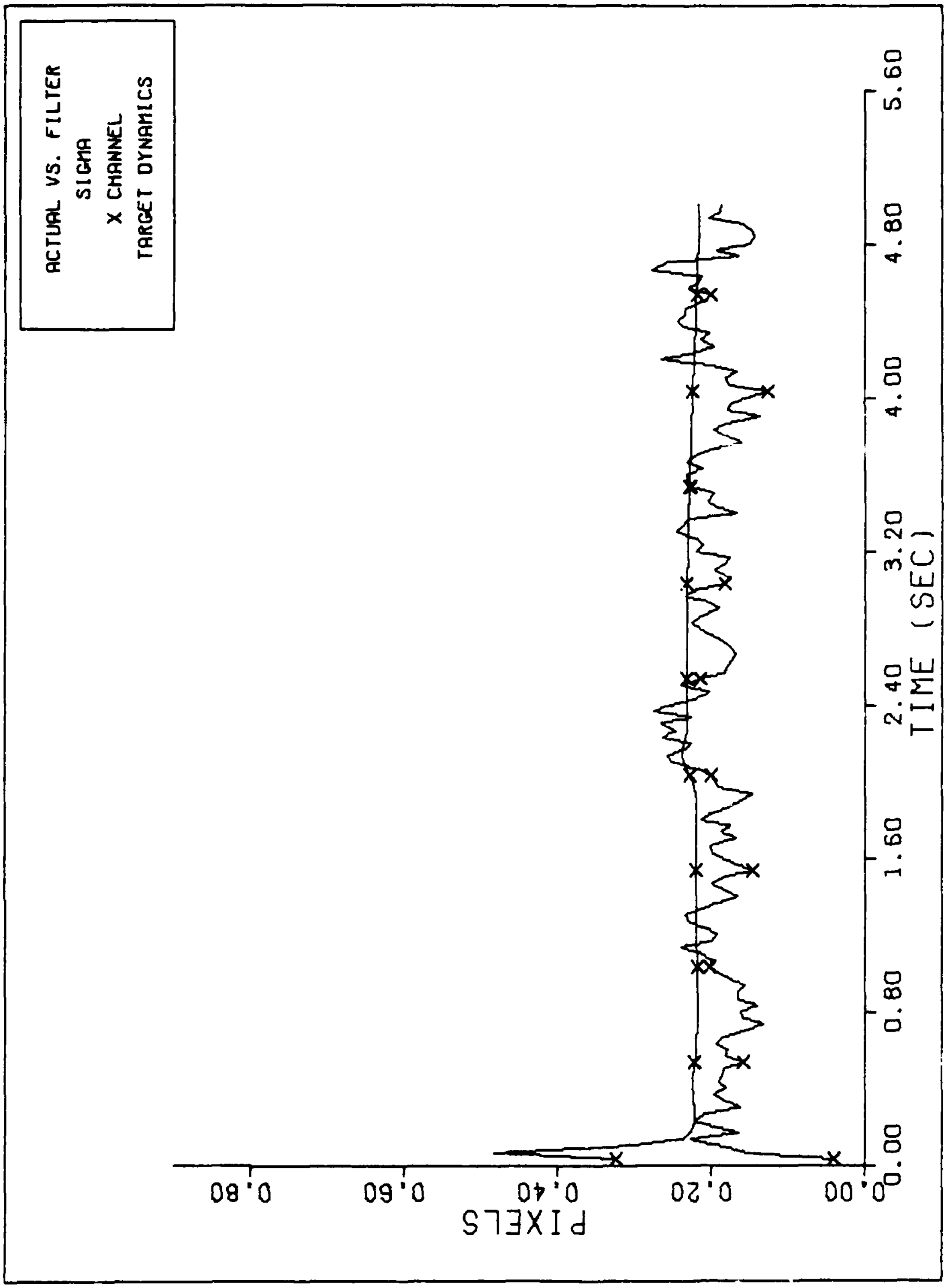
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-4d



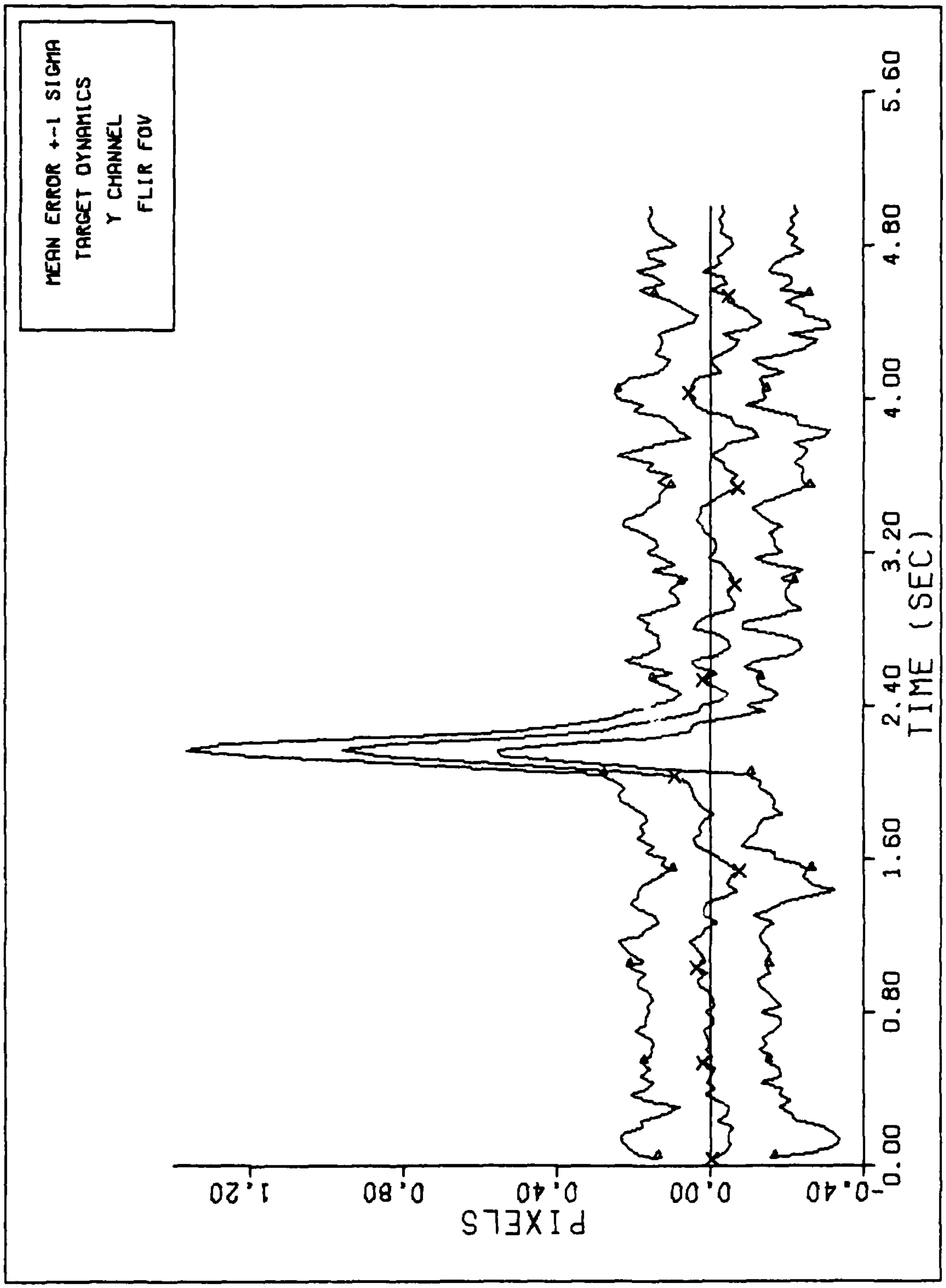
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure F-5a



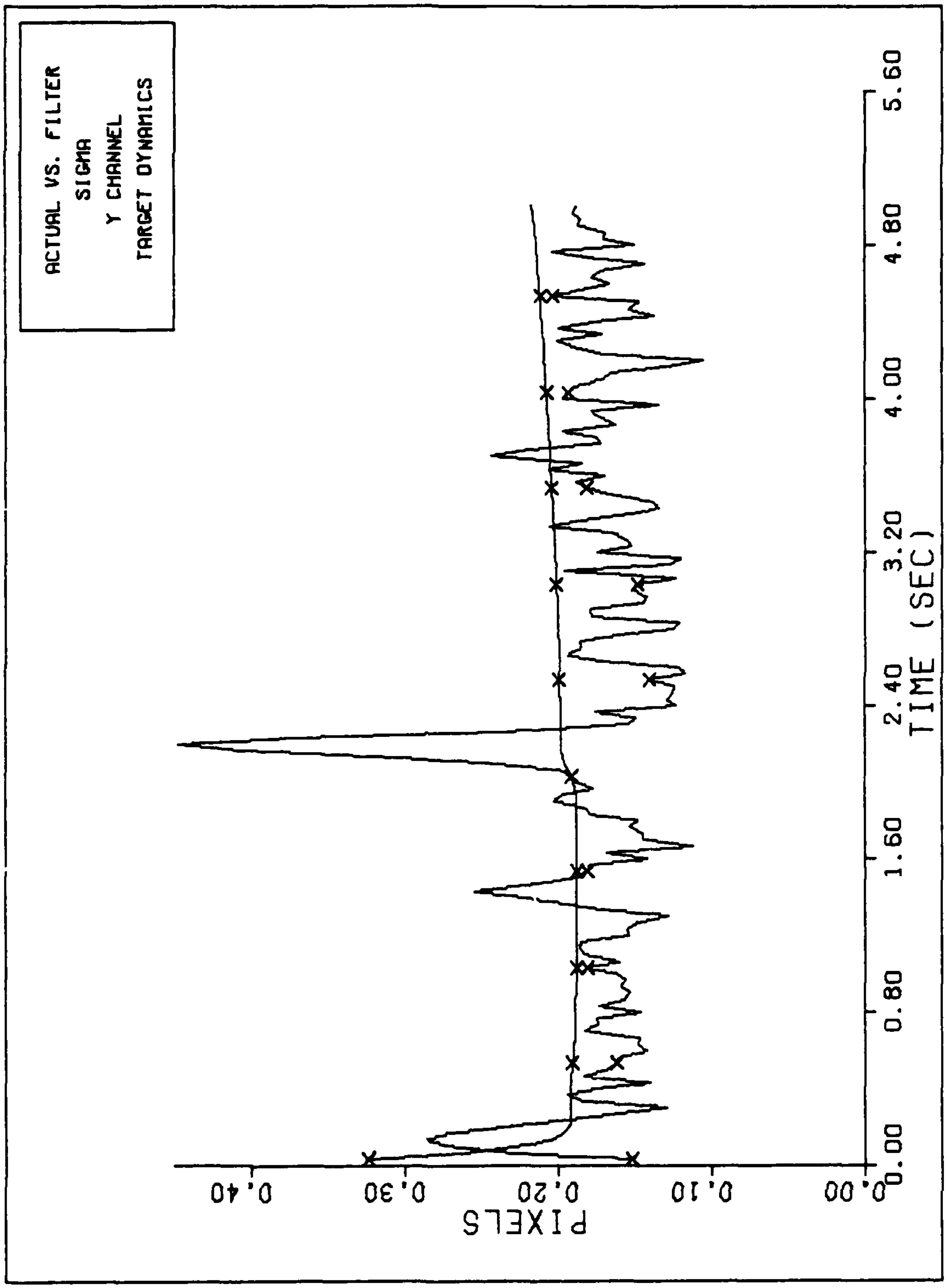
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-5b



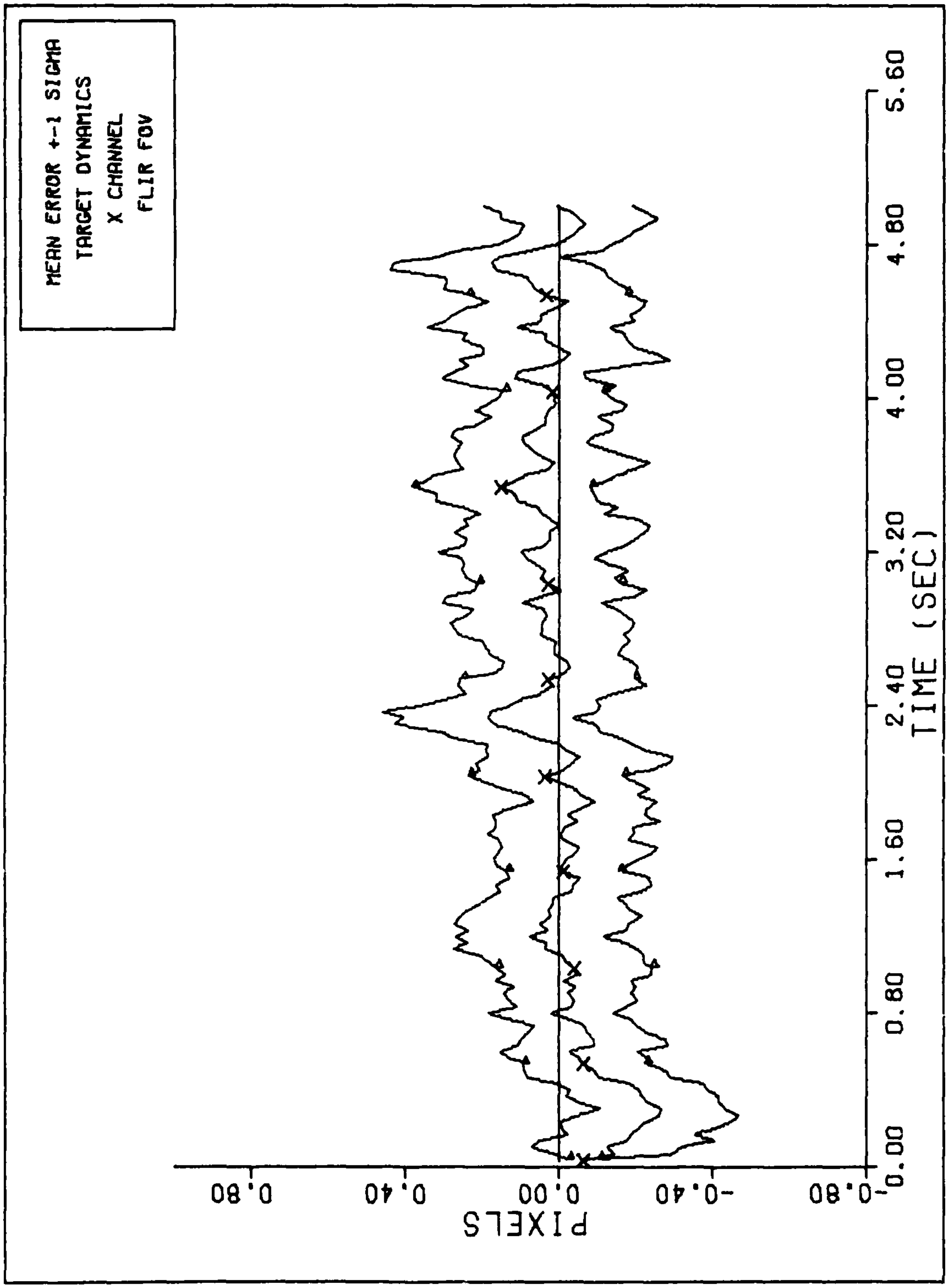
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure F-5c



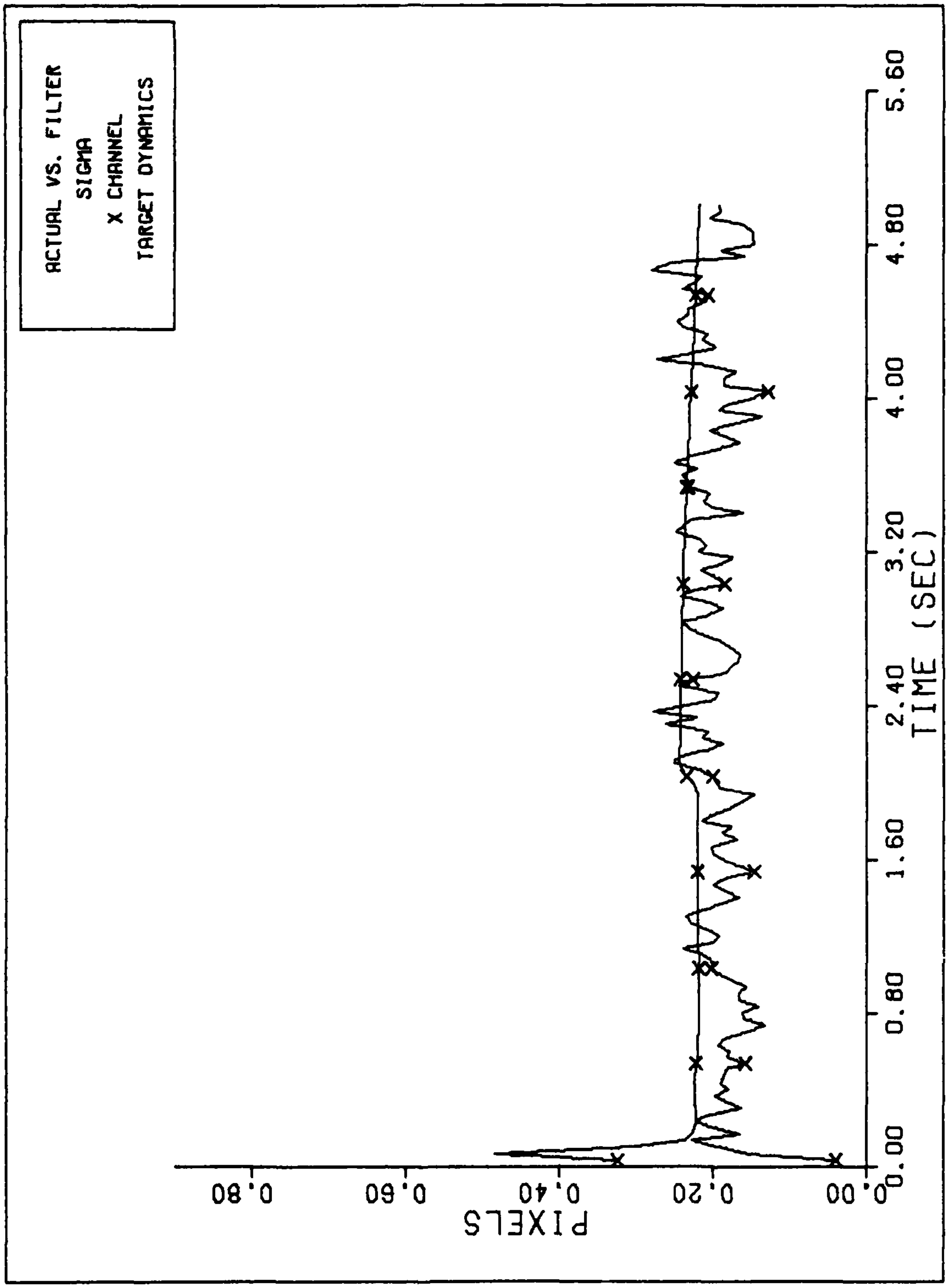
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-5d



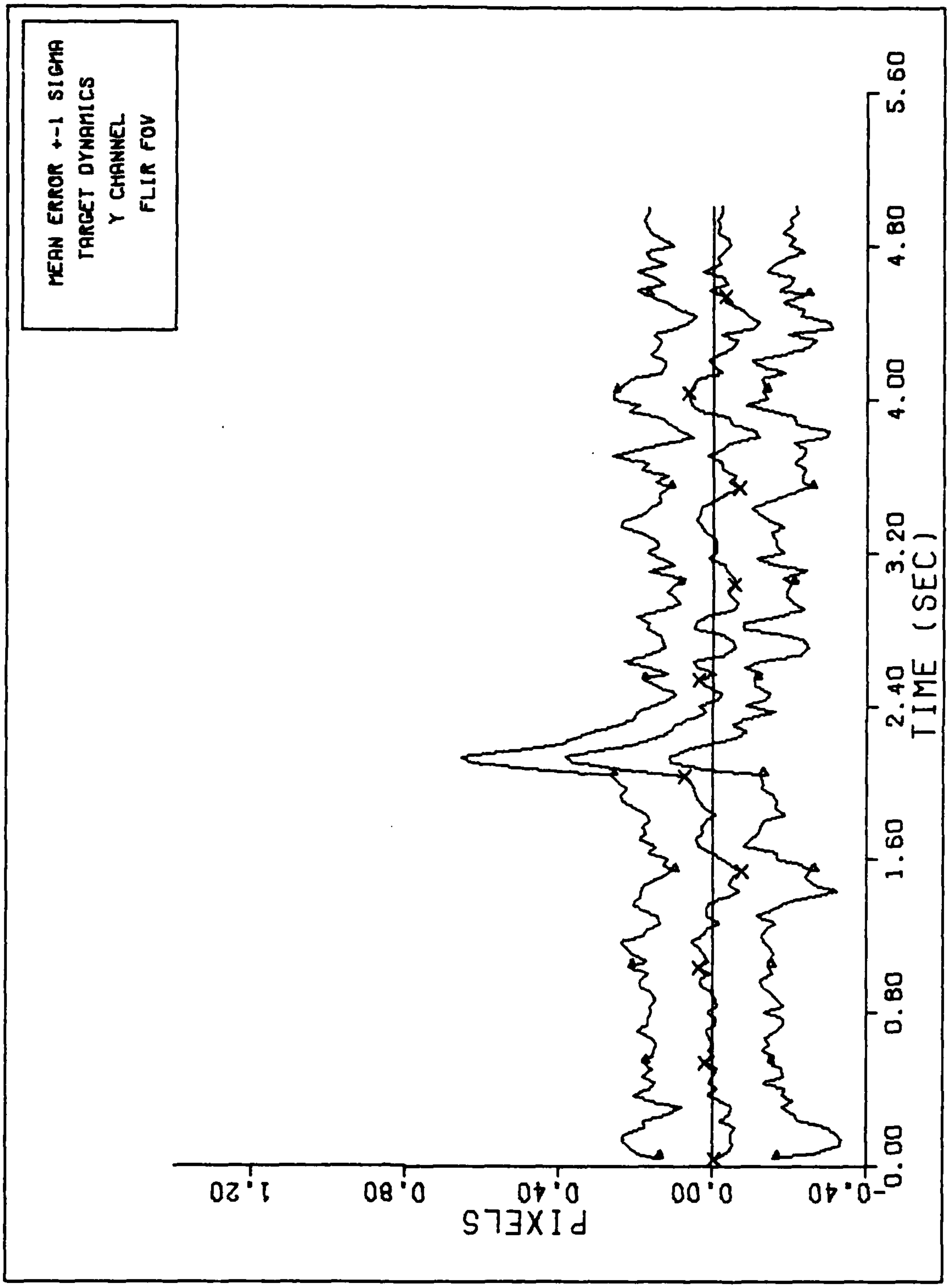
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure F-6a



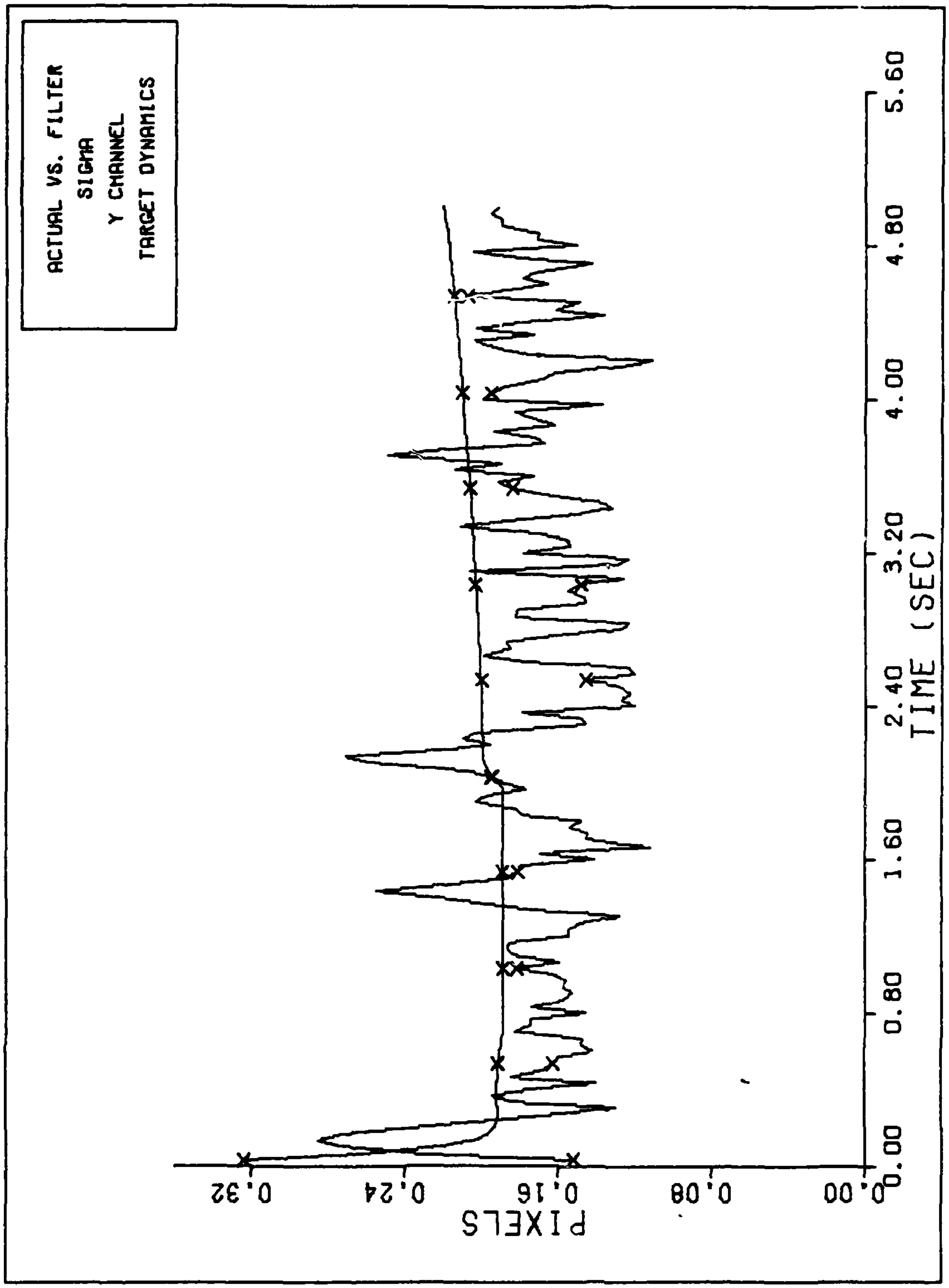
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-6b



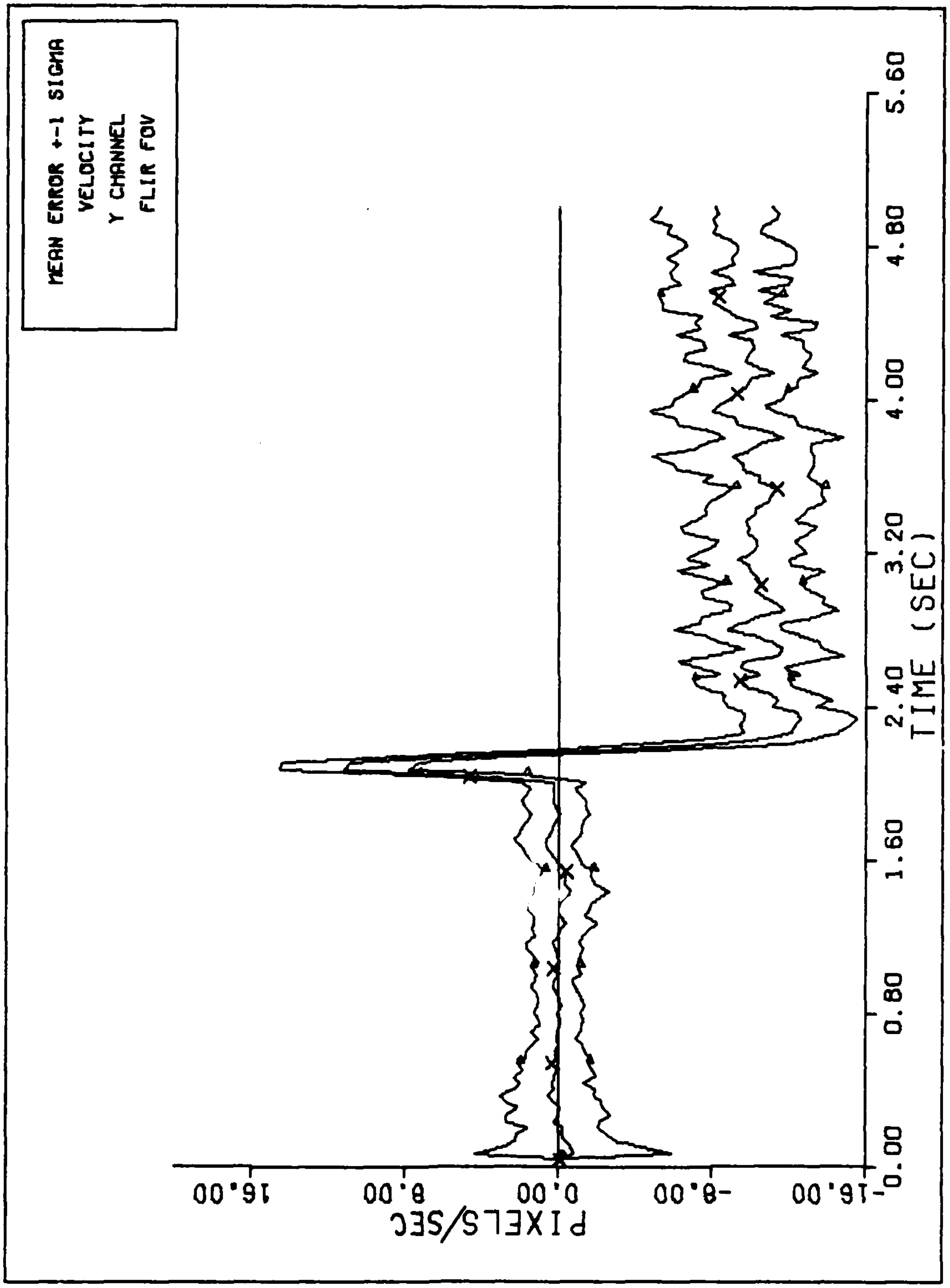
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure F-6c



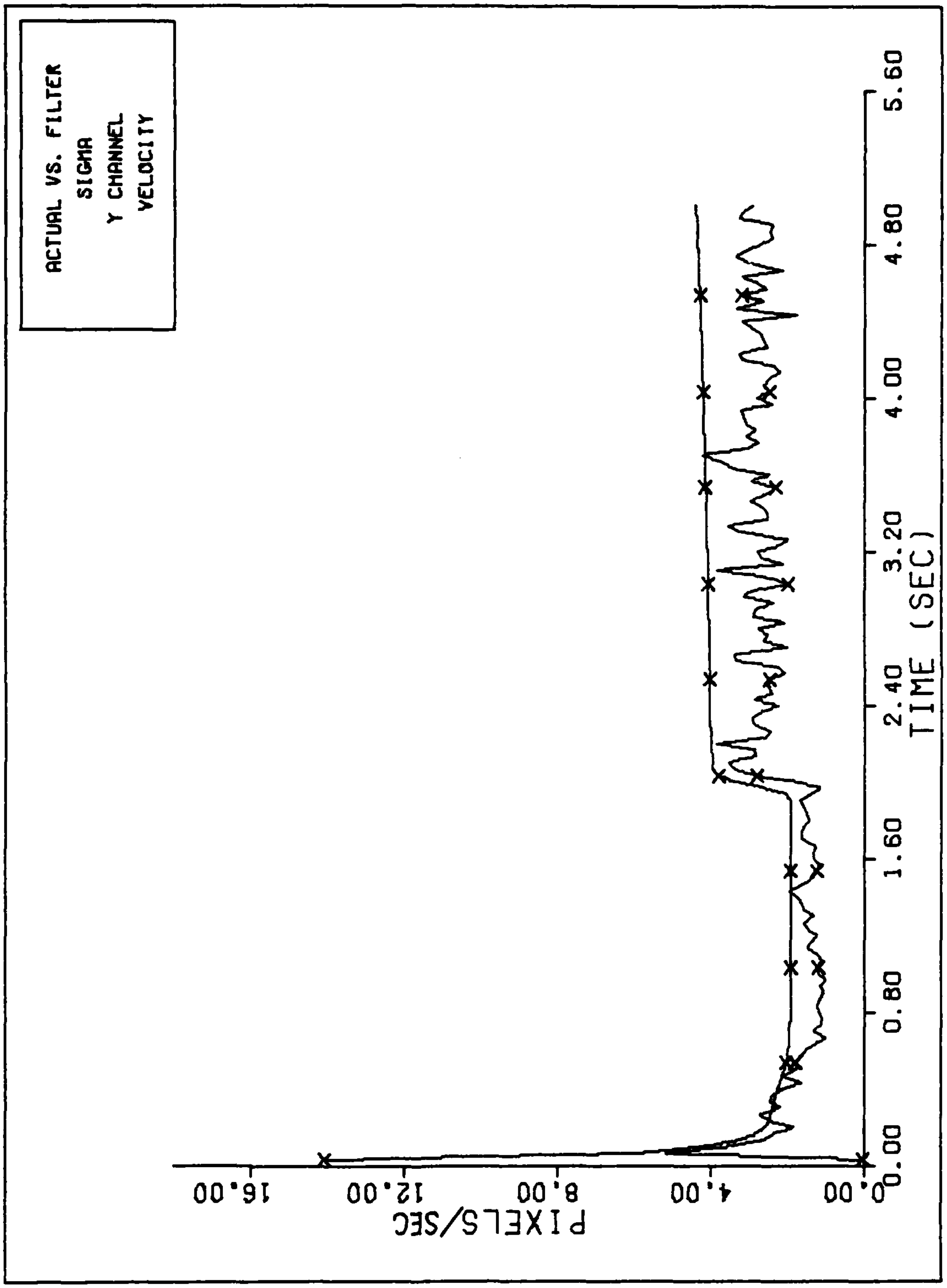
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-6d



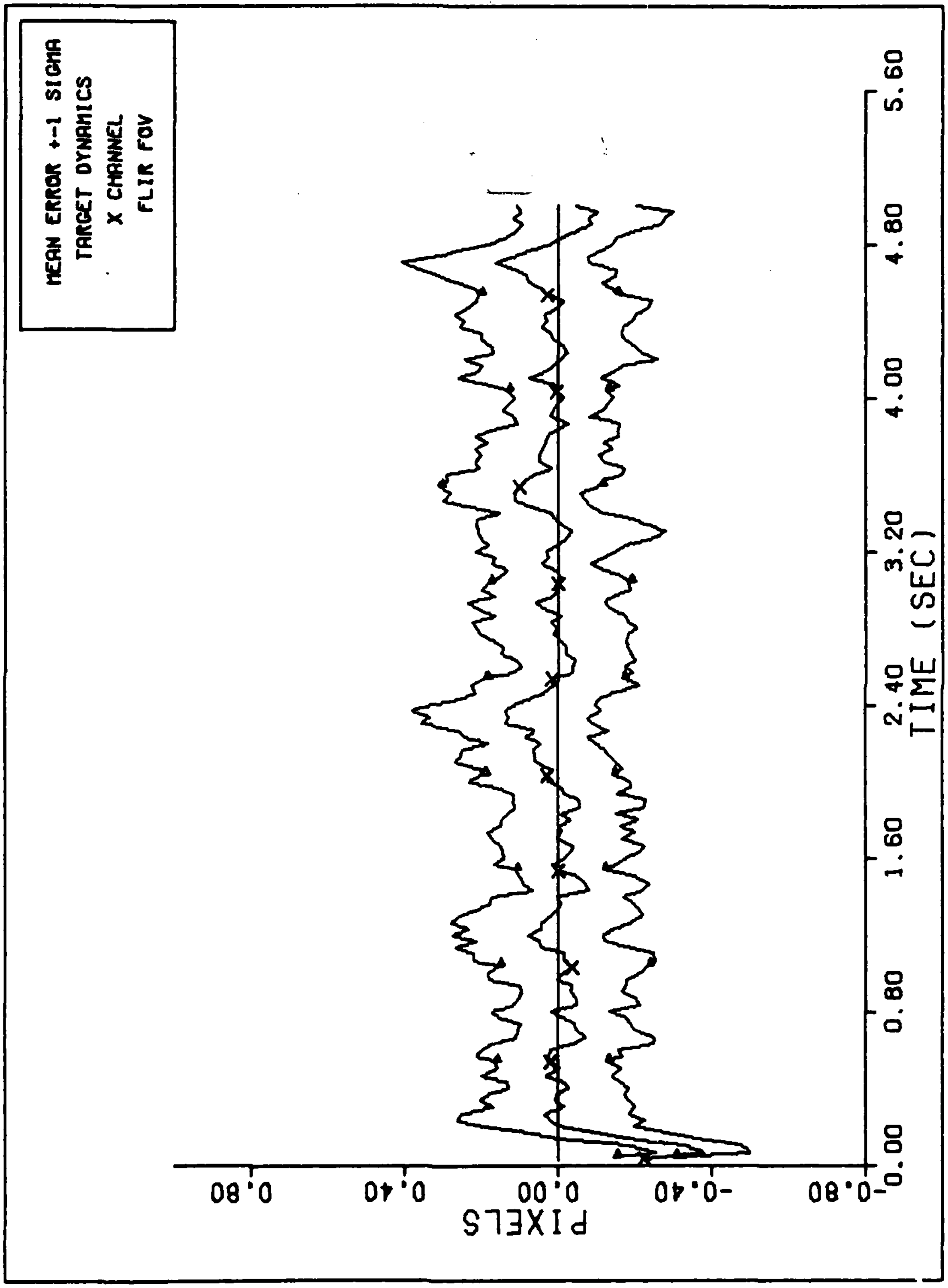
Y CHANNEL VELOCITY ERROR (S/N=12.5)

Figure F-6e



FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

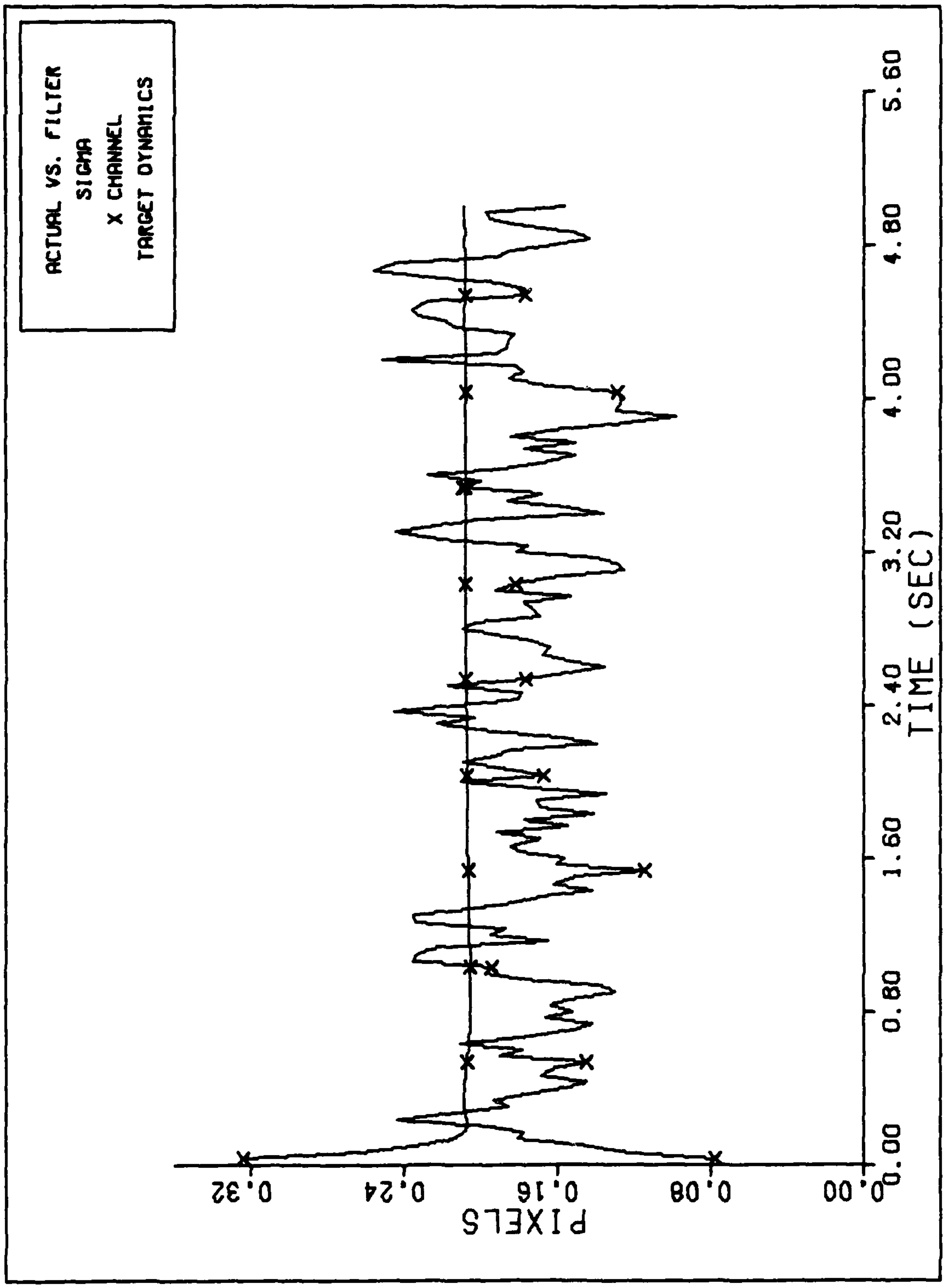
Figure F-6f



X CHANNEL DYNAMICS ERROR (S/N=12.5)

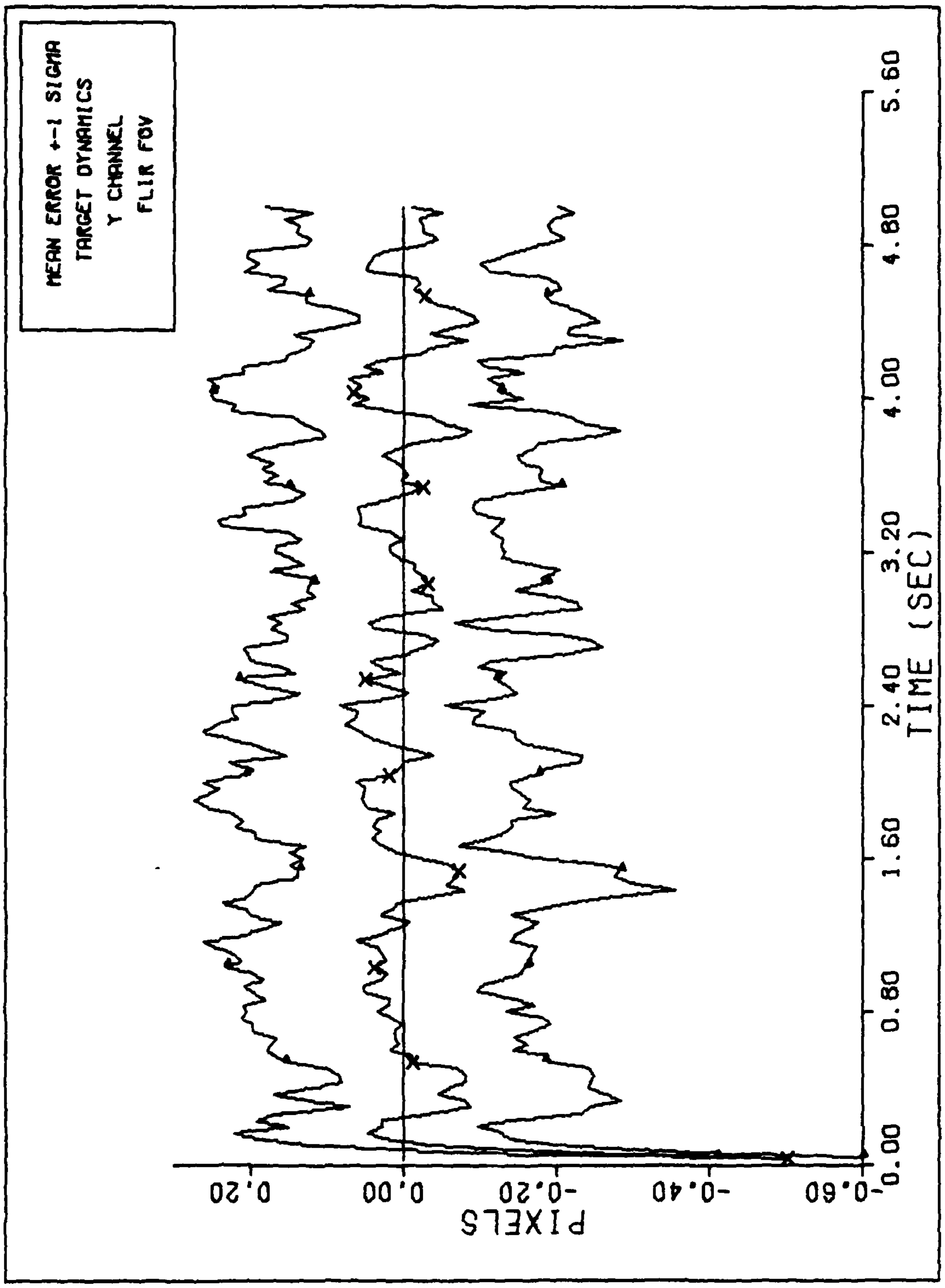
Figure F-9a

F9a



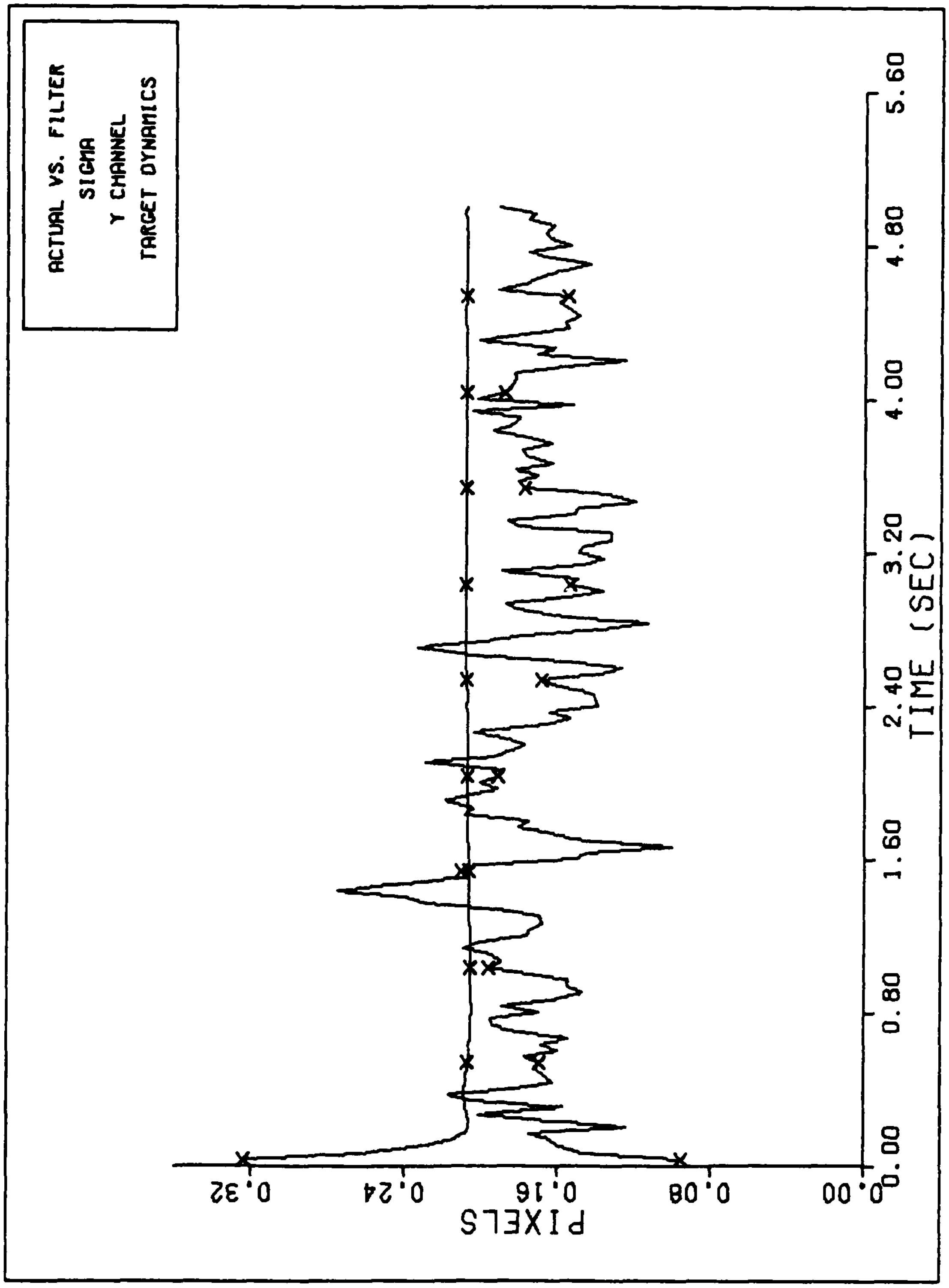
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-9b



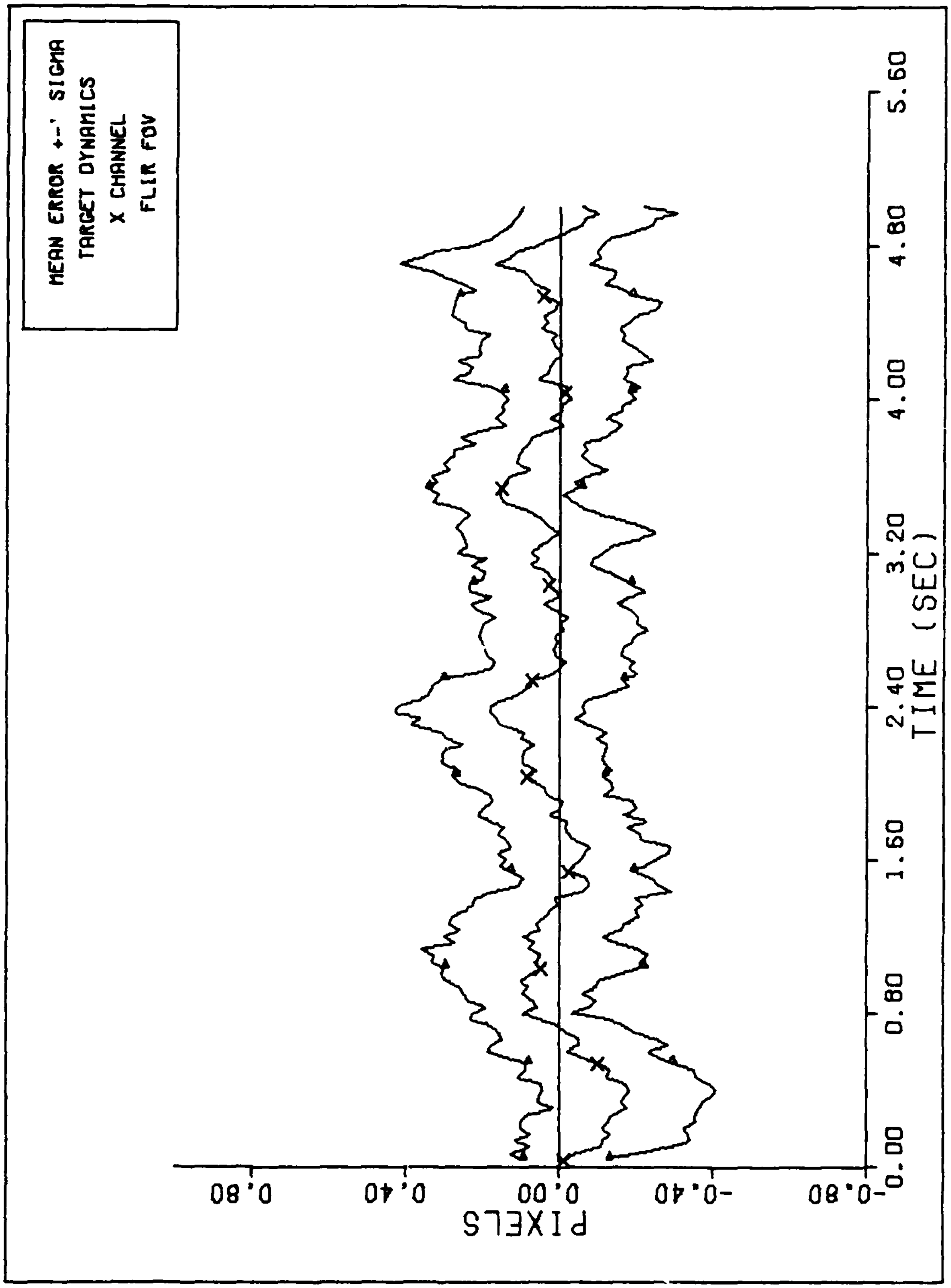
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure F-9c



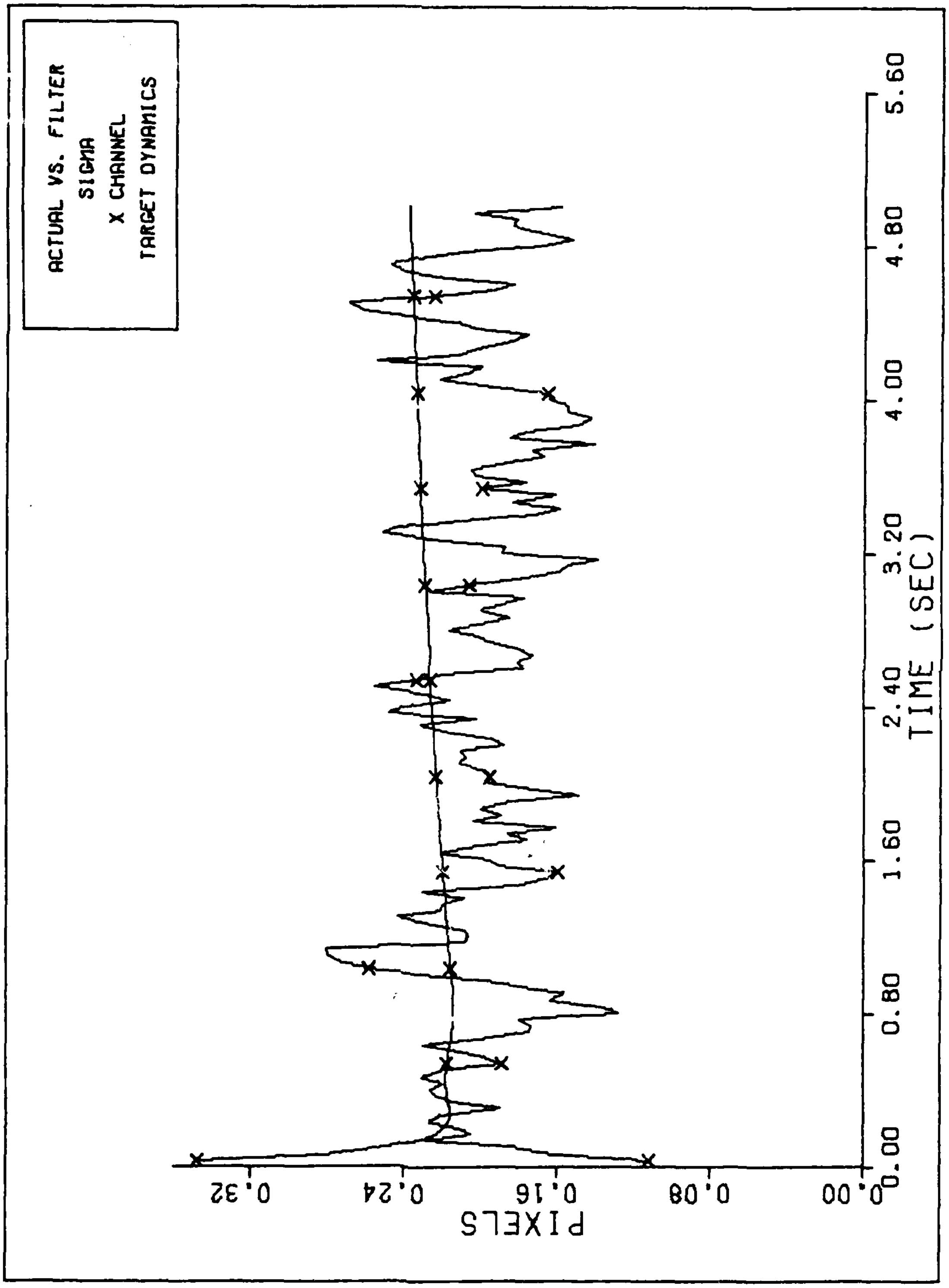
FILTER VS. ACTUAL SIGMA PLOT (S/N =12.5)

Figure F-9d



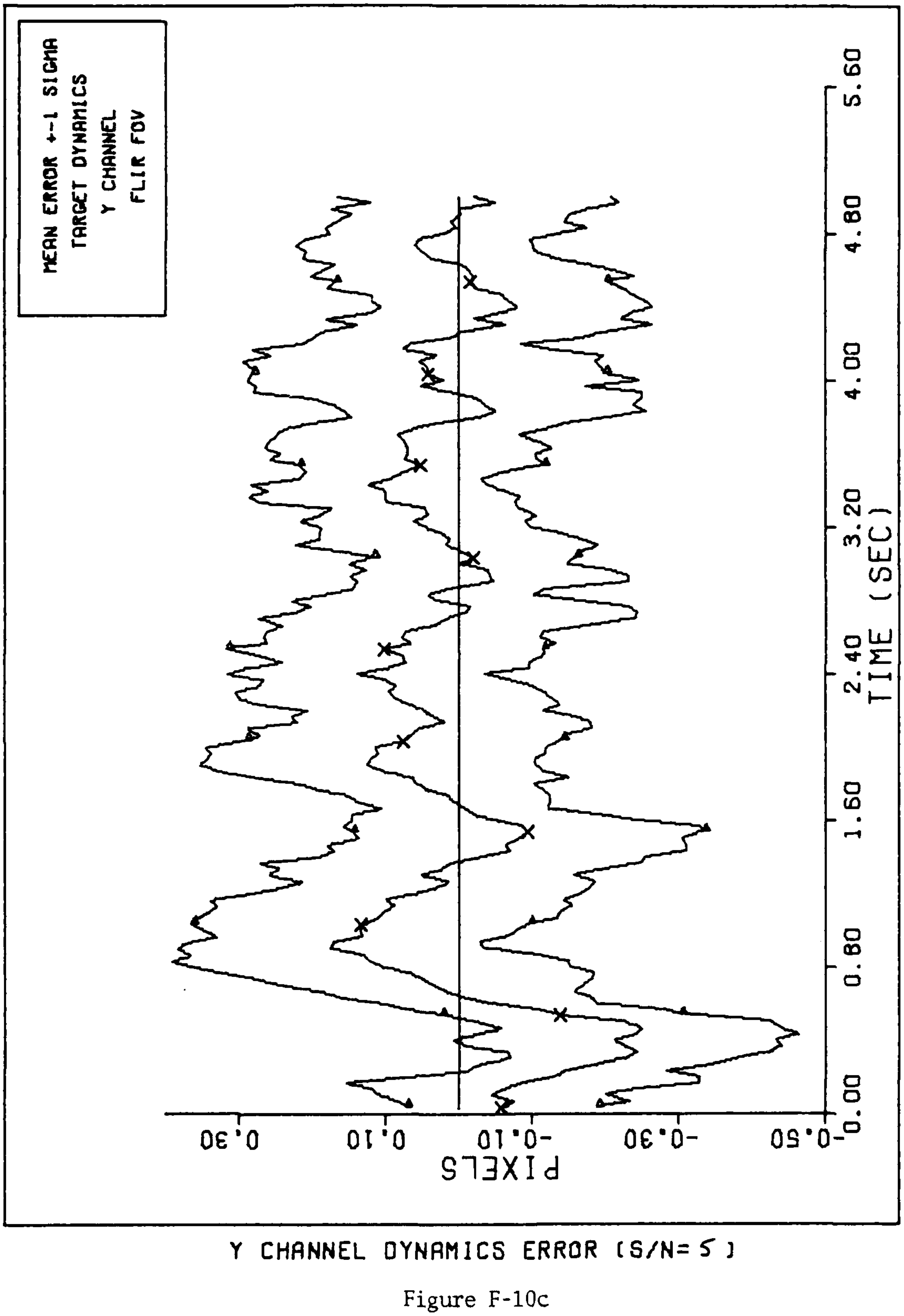
X CHANNEL DYNAMICS ERROR (S/N= 5)

Figure F-10a



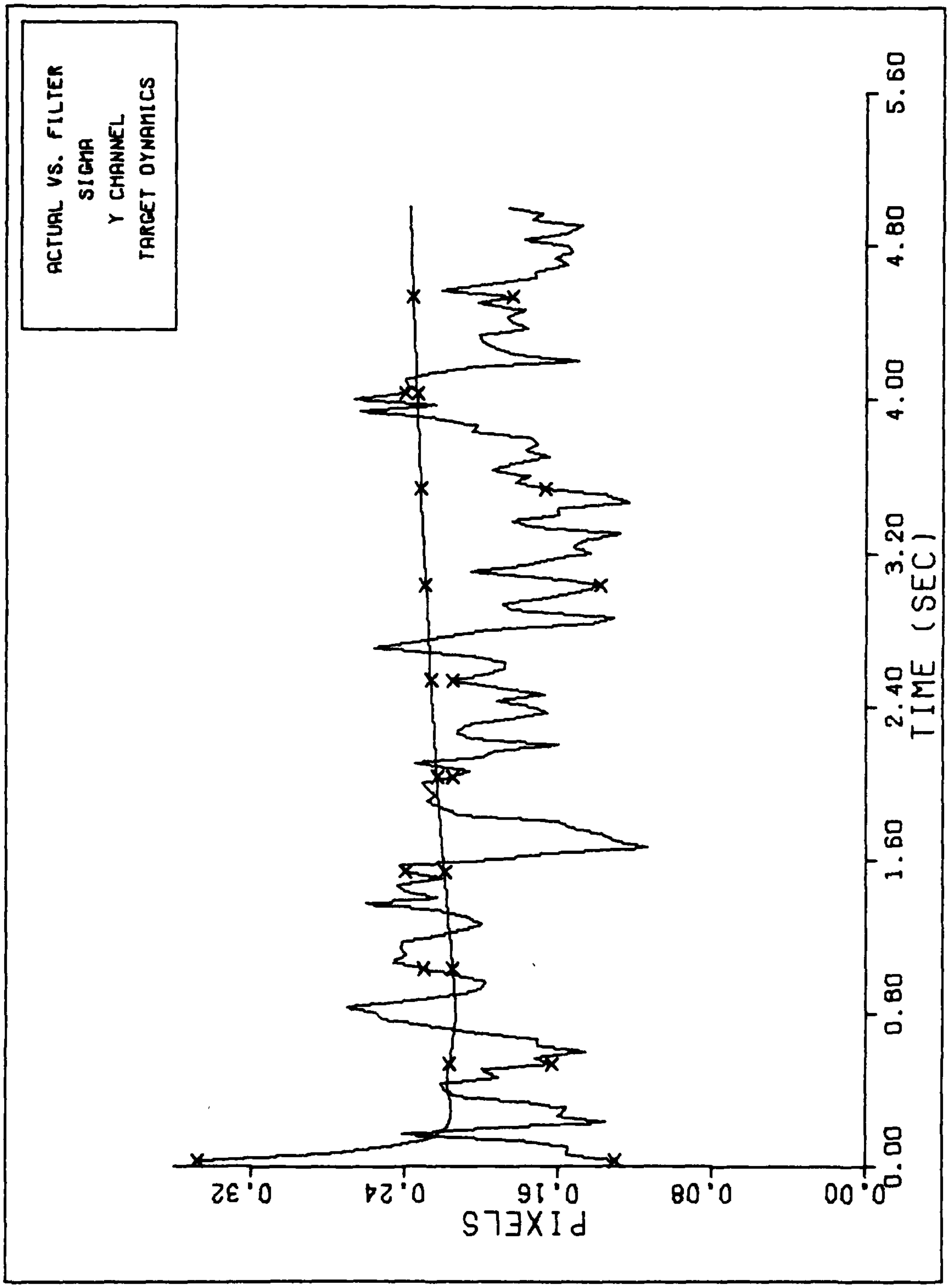
FILTER VS. ACTUAL SIGMA PLOT (S/N = 5)

Figure F-10b



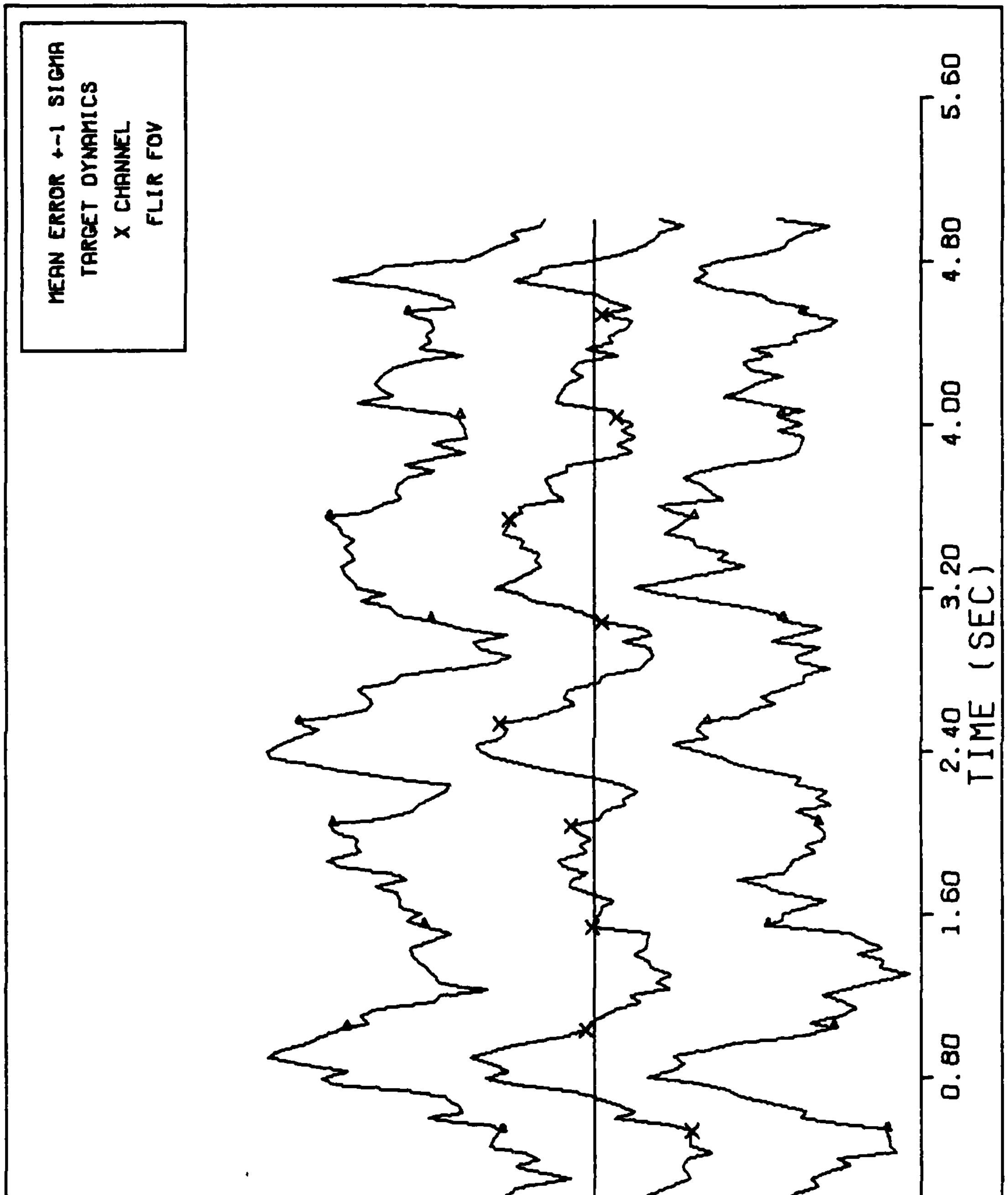
Y CHANNEL DYNAMICS ERROR (S/N= 5)

Figure F-10c



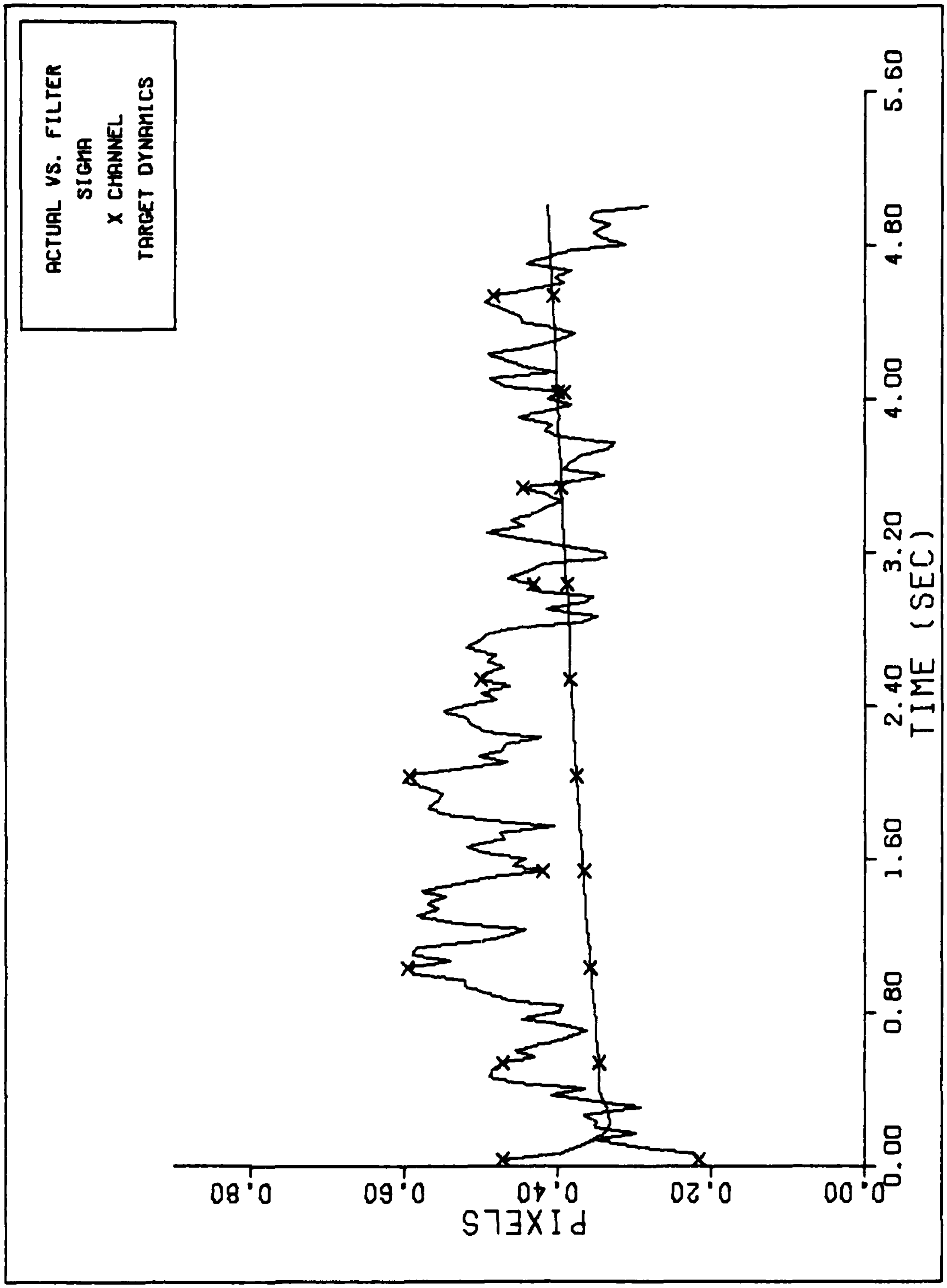
FILTER VS. ACTUAL SIGMA PLOT (S/N = 5)

Figure F-10d



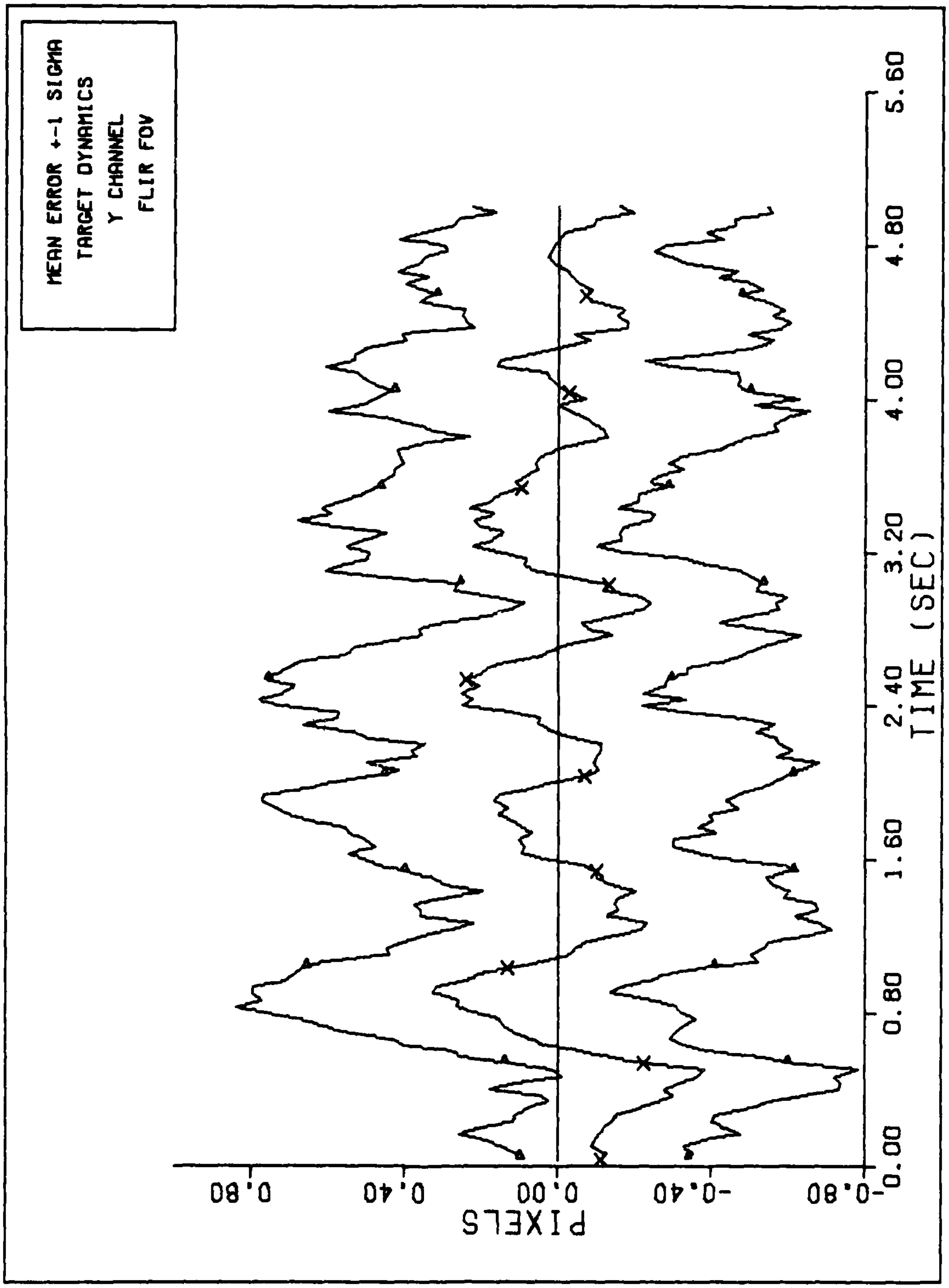
X CHANNEL DYNAMICS ERROR (S/N= 2)

Figure F-11a



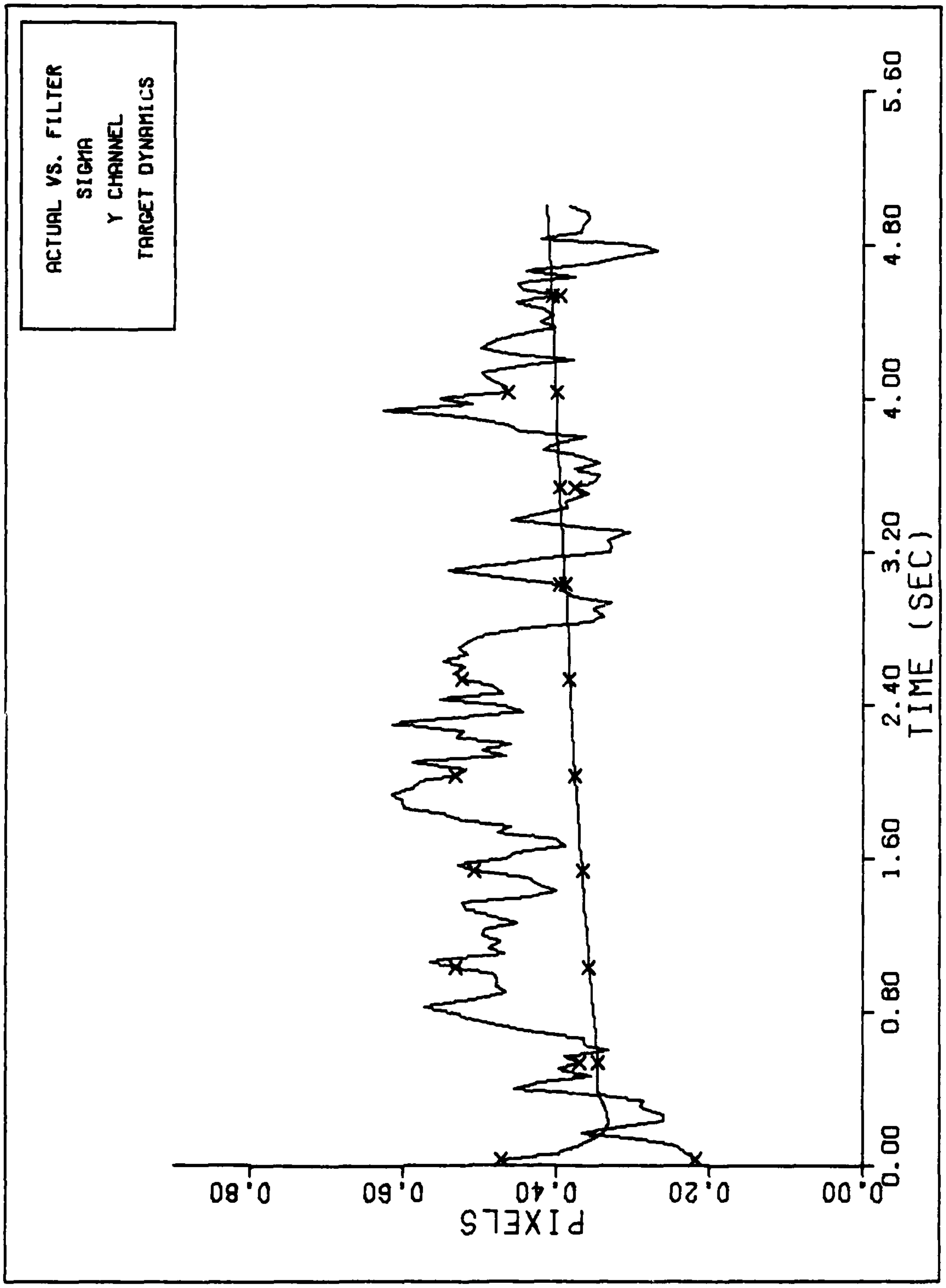
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure F-11b



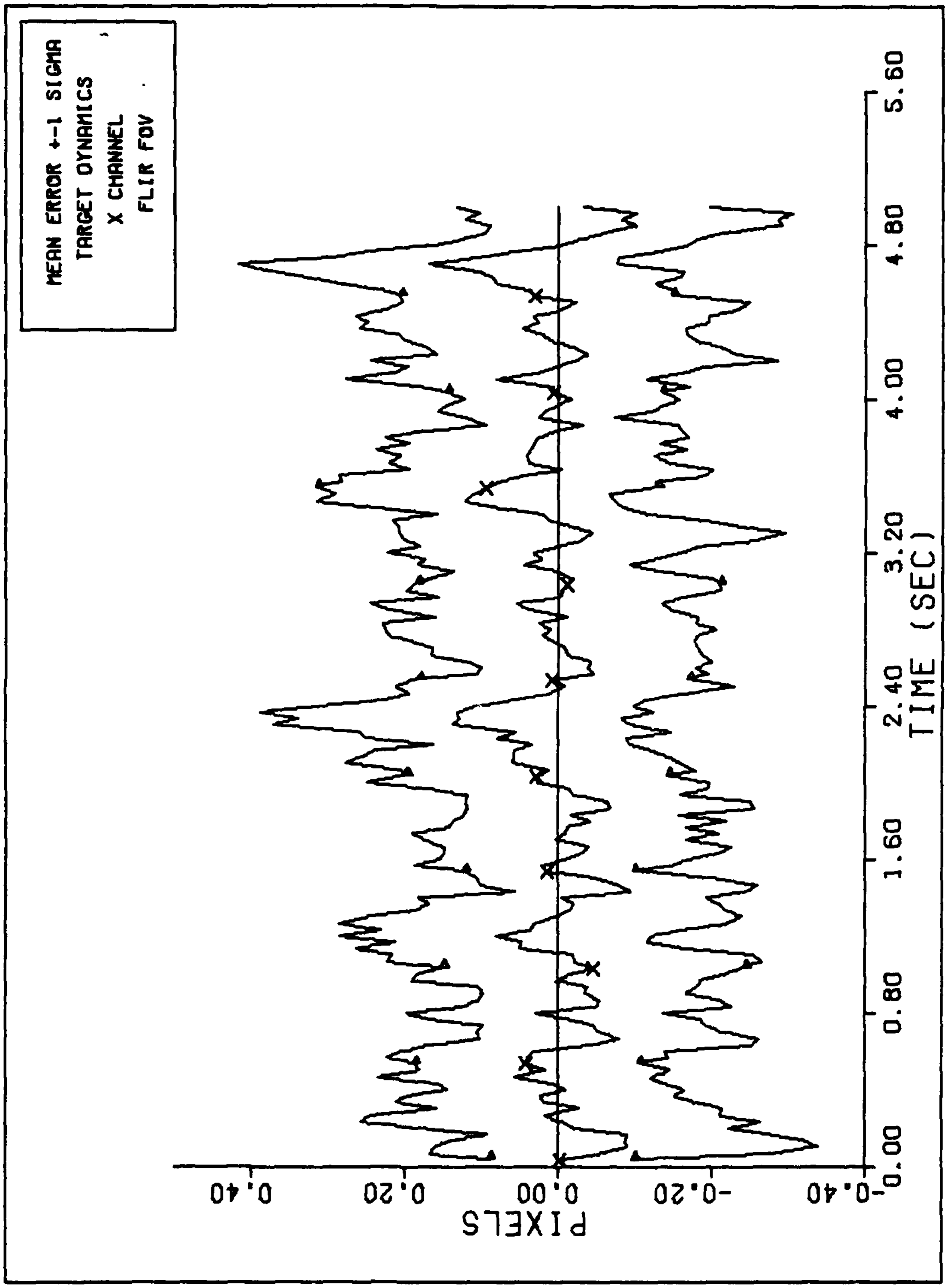
Y CHANNEL DYNAMICS ERROR (S/N= 2)

Figure F-11c



FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

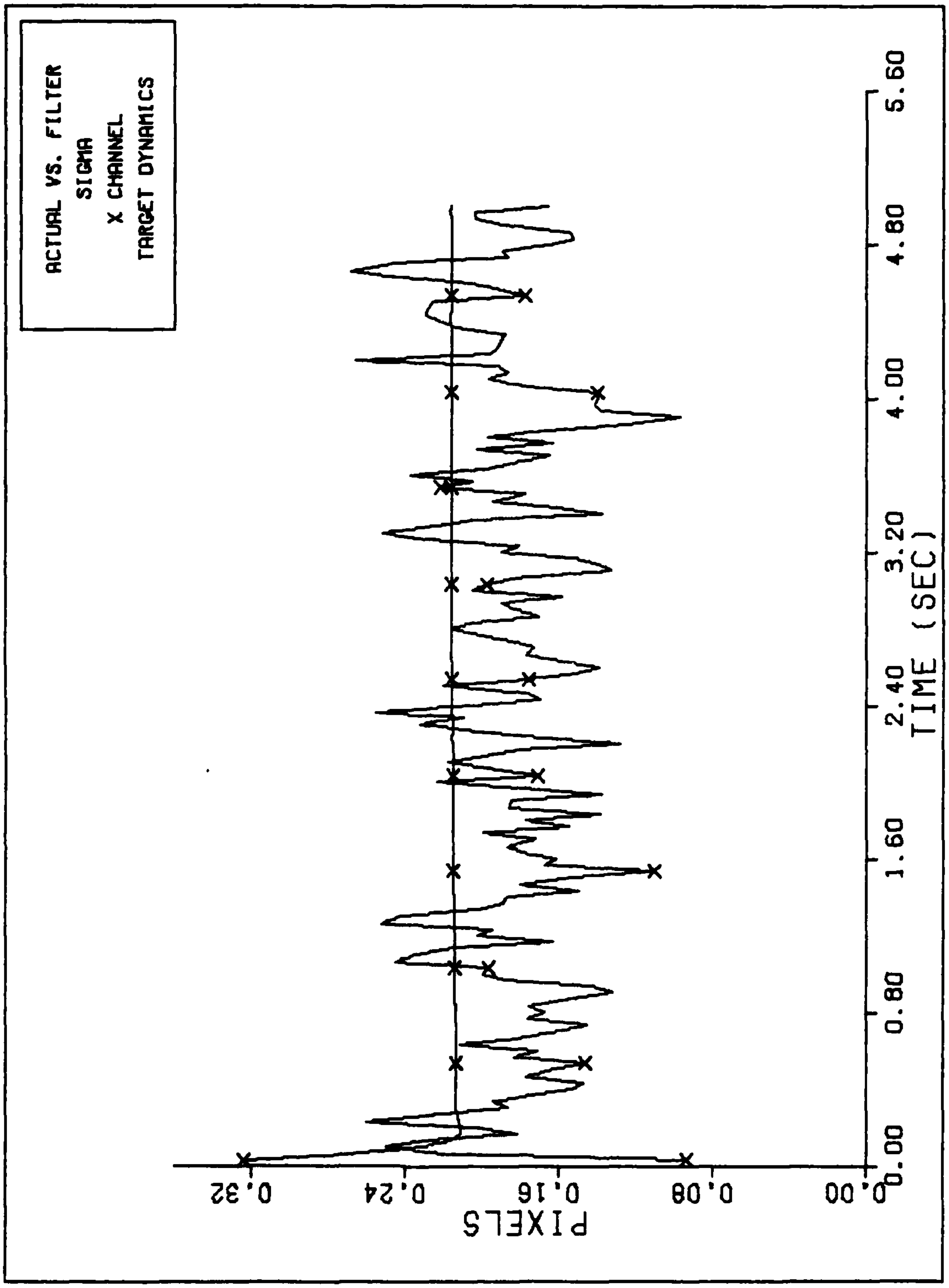
Figure F-11d



X CHANNEL DYNAMICS ERROR (S/N= 2)

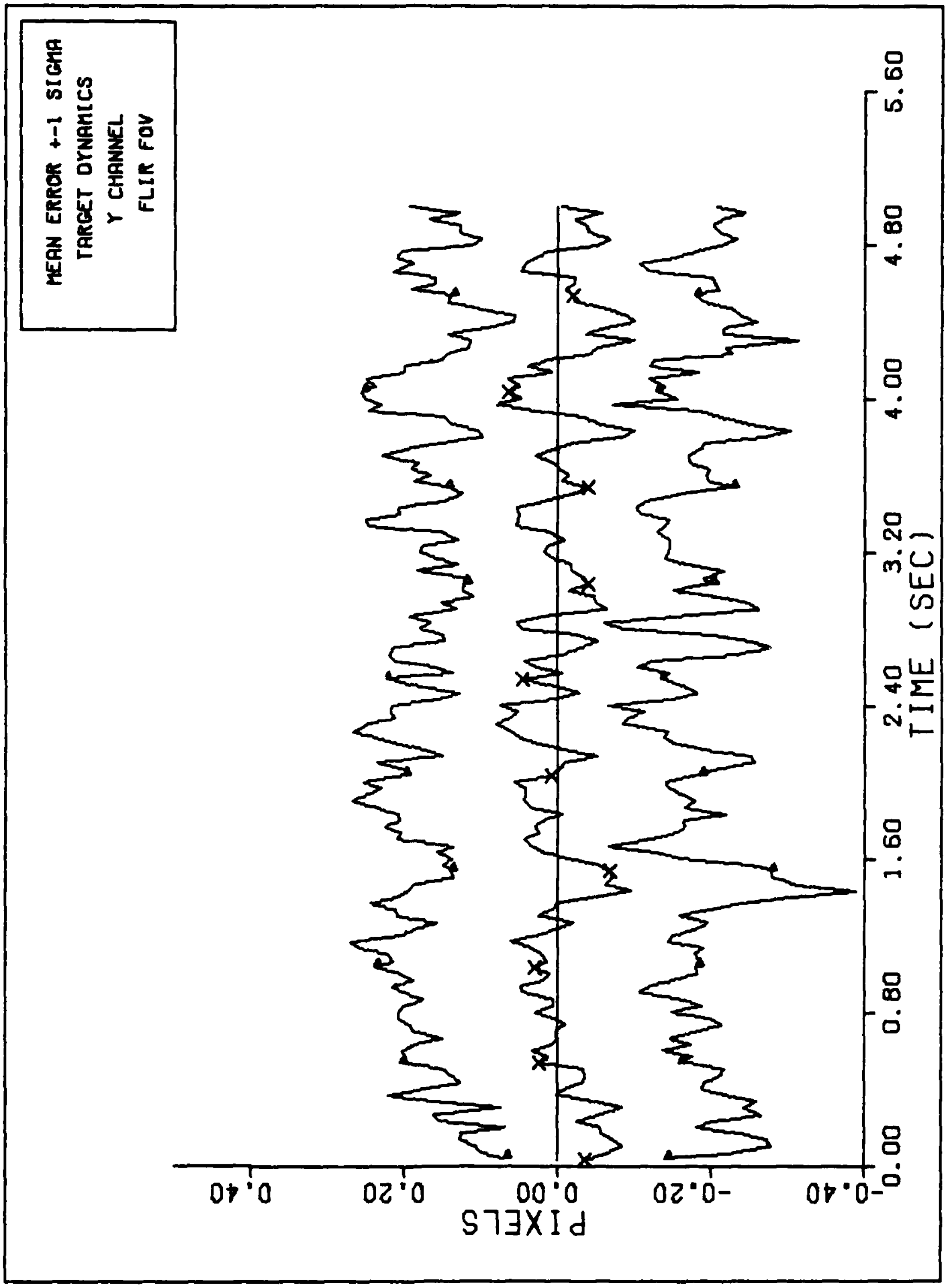
Figure F-12a

F12a



FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

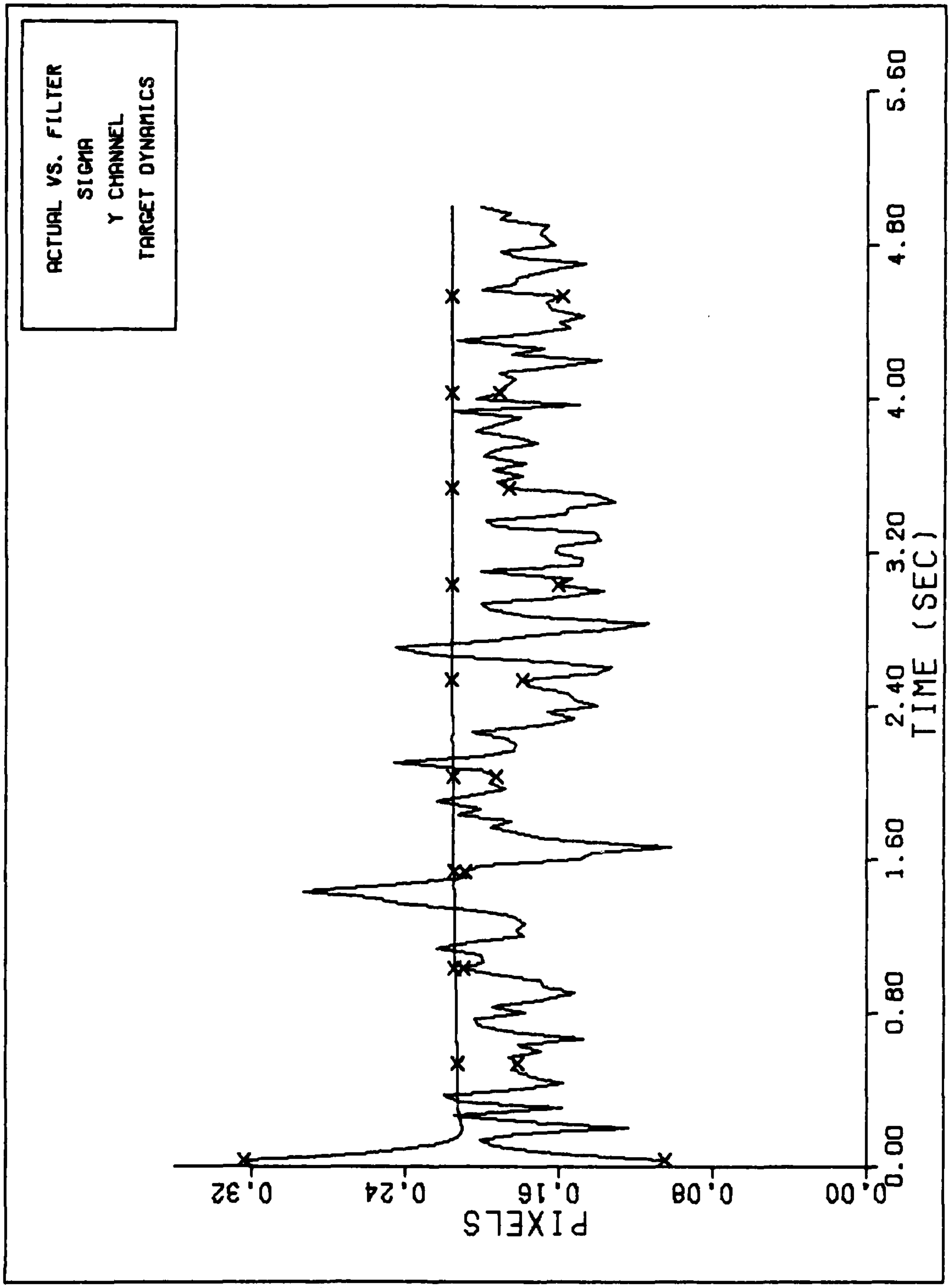
Figure F-12b



Y CHANNEL DYNAMICS ERROR (S/N= 2)

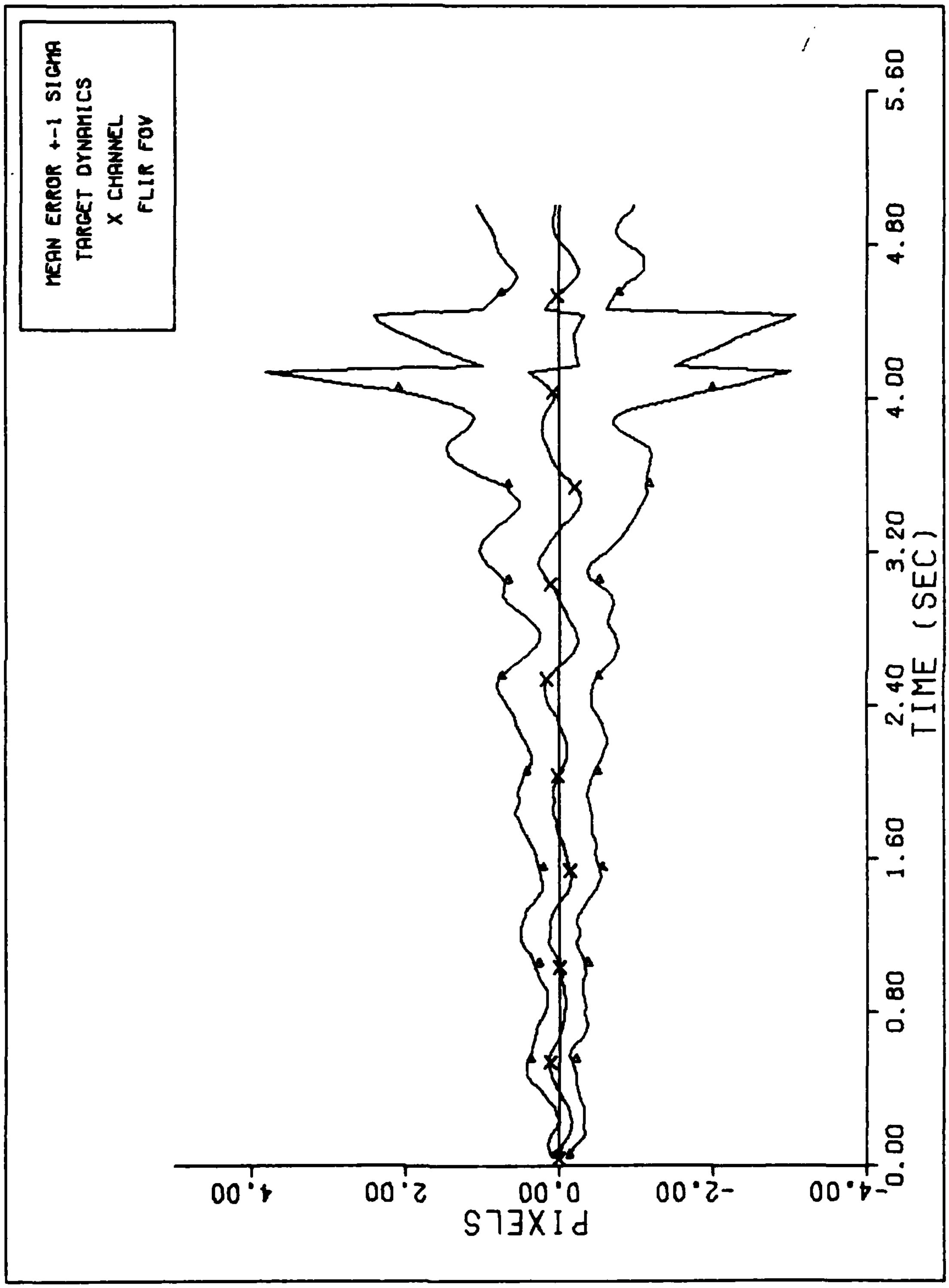
Figure F-12c

F12c



FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

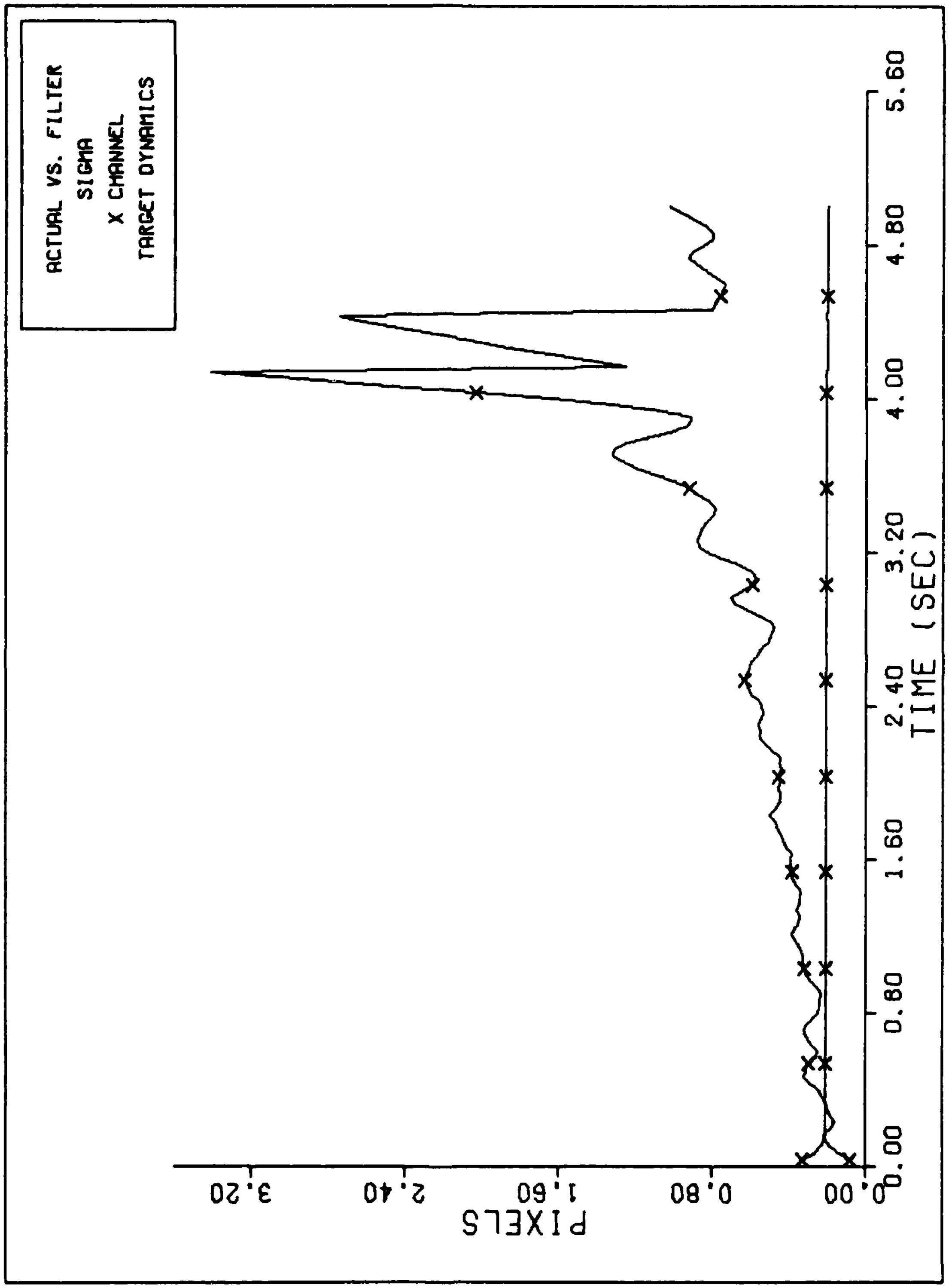
Figure F-12d



X CHANNEL DYNAMICS ERROR (S/N= 5)

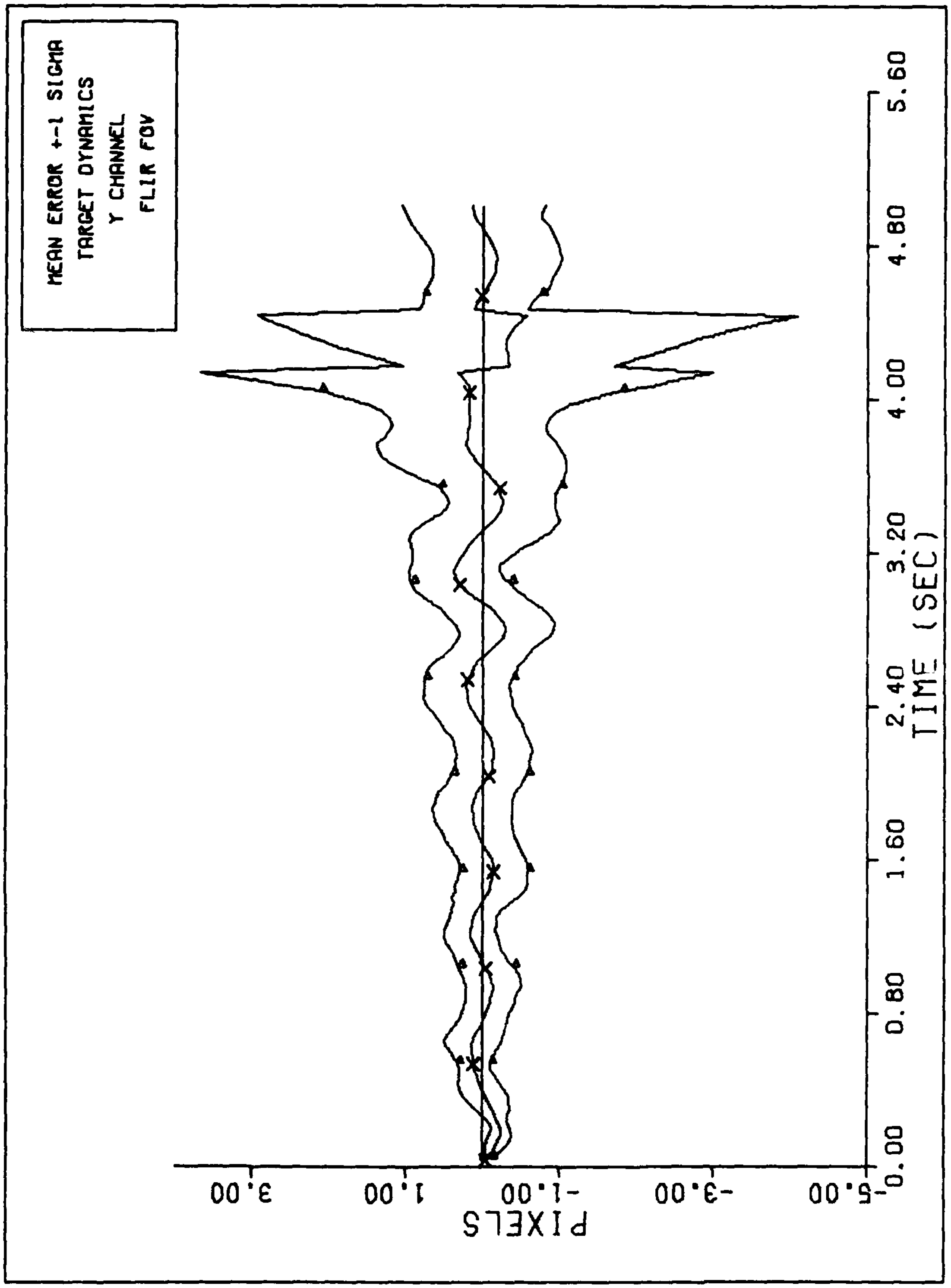
Figure F-13a

F13a



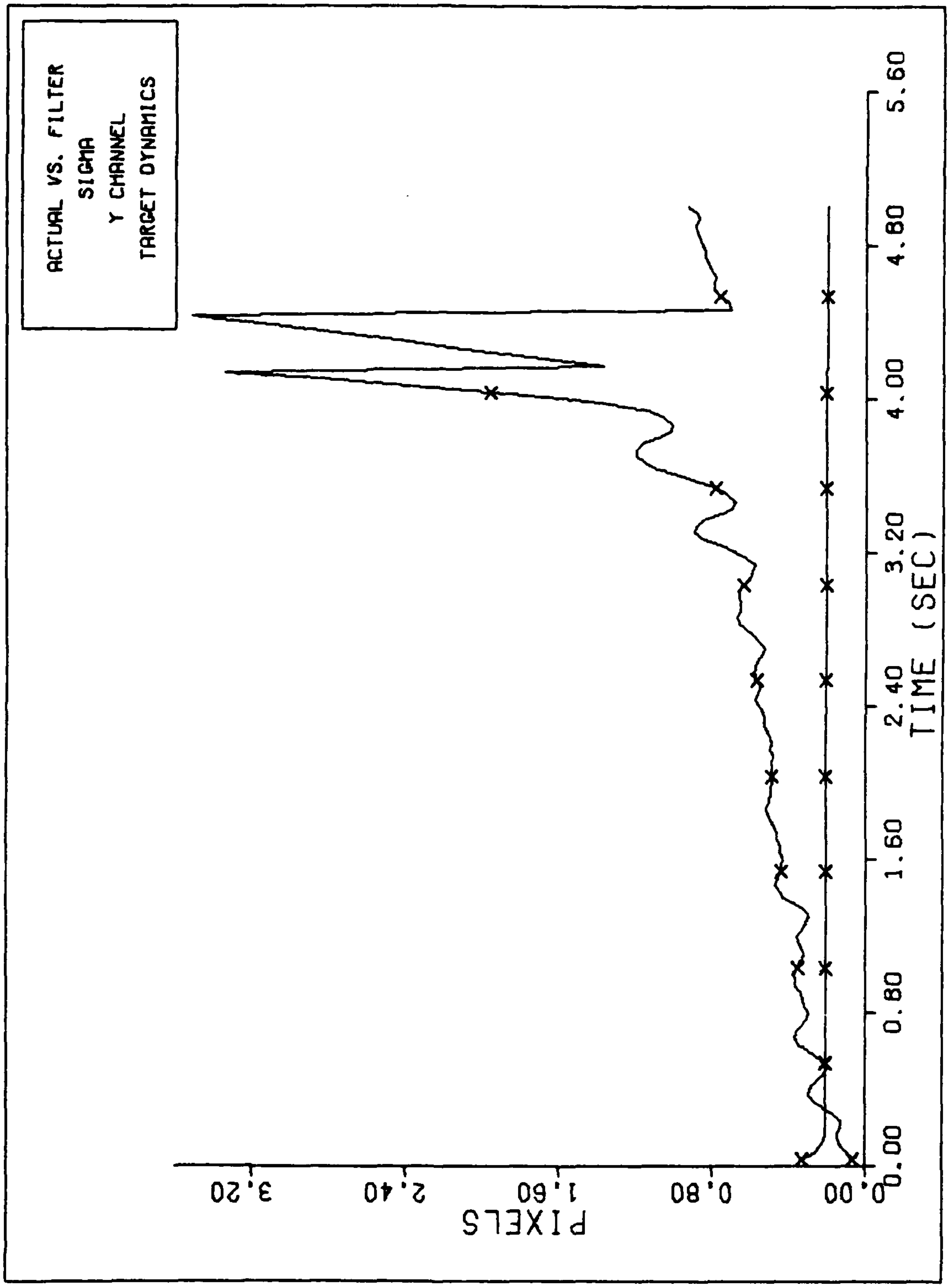
FILTER VS. ACTUAL SIGMA PLOT (S/N = 5)

Figure F-13b



Y CHANNEL DYNAMICS ERROR (S/N= 5)

Figure F-13c



FILTER VS. ACTUAL SIGMA PLOT (S/N = 5)

Figure F-13d

Appendix G

Appendix G

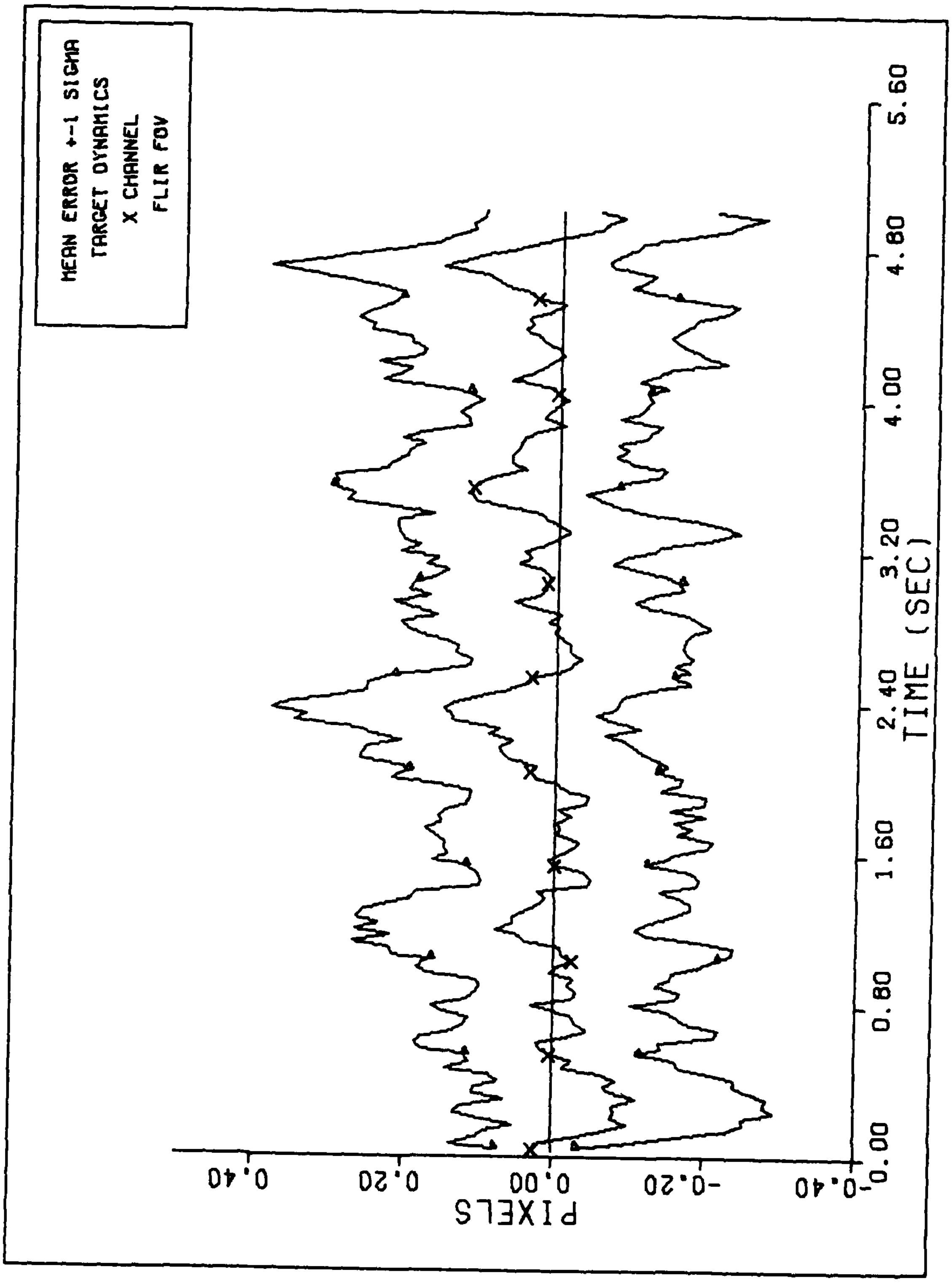
Performance Plots for the Constant Gain Extended Kalman Filter

Constant Parameters

Symbol	FORTRAN Code	Valve
σ_D	SIGSI	0
σ_f	SIGFLR	0
σ_A	SIGAT	0.2
σ_{g_2T}	SIGMS	1
AR_T	ASPRO	5
	ISPTL	'NO'
	NRUN	20
	TFINAL	5
$\sigma_{\dot{x}}$	SIGMFO	-3
AR_F	ARO	1
σ_{AF}	SIGF2	0.2
σ_v	RF	2

Input Parameters

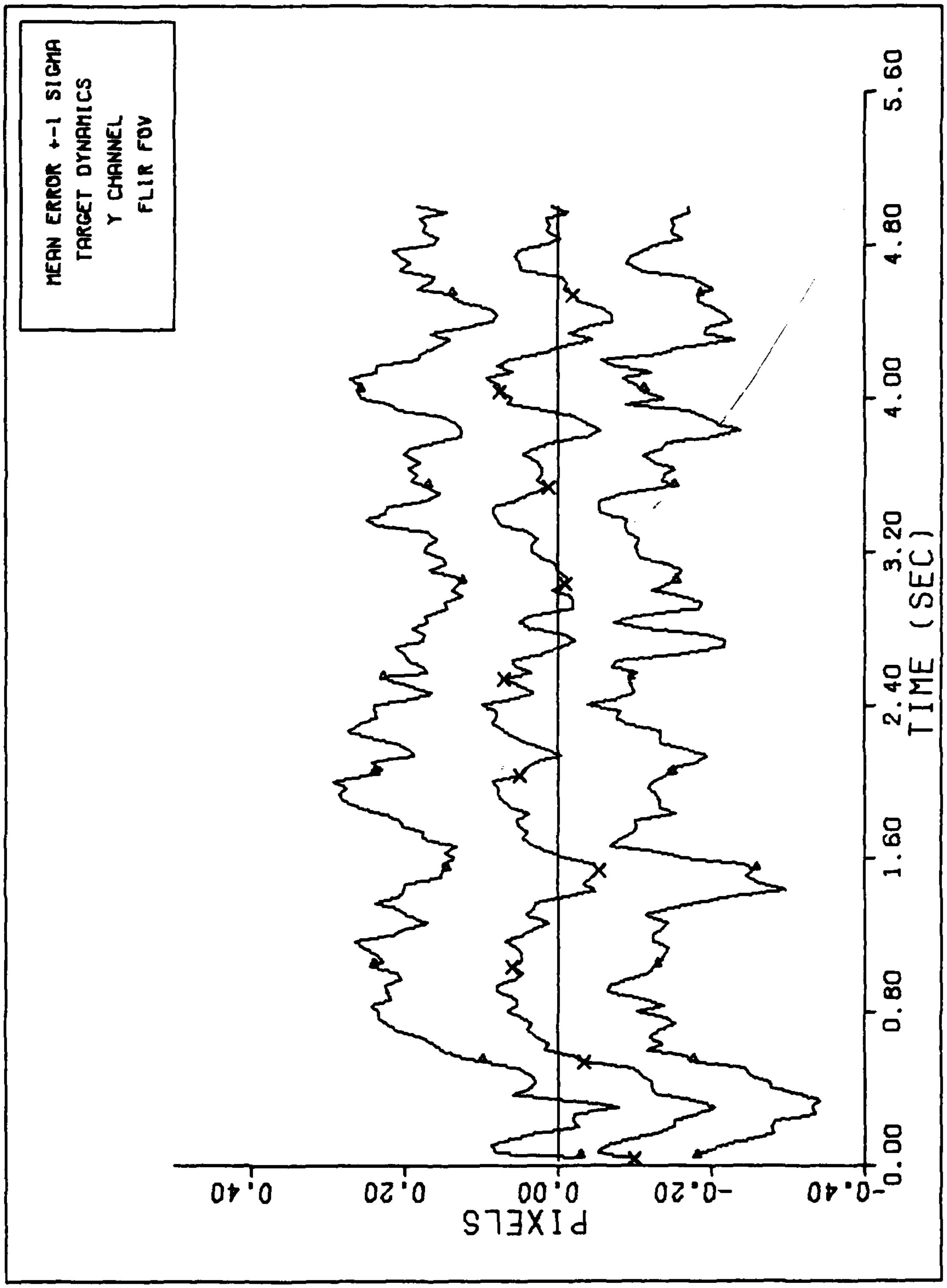
Symbol	Fortran Code / Case	G1	G2	G3	G4	G5	G6	G7
I_{max}	IMAX	25	25	25	25	25	25	10
σ_N	SIGMAB	2	2	2	2	2	2	2
$I_{max}F$	FIMAX0	25	25	25	25	25	25	20
DF	SIGF10	-	-	1000	20,000	20,000	-	-
Acquisition Q_{FD}		-	-	NO	NO	NO	-	-
$\underline{K}_{ss}$	GAIN	Entire Run	Entire Run	t_i 2.0	t_i 2.0	t_i 2.0	Entire Run	Entire Run
S/N	SN	12.5	12.5	12.5	12.5	12.5	12.5	5
Turn (G)		0	5	5	5	10	0	0
Initial Conditions $x_d(0)$	XFP(1)	0	0	0	0	0	0.5	0
$y_d(0)$	SFP(2)	0	0	0	0	0	0.5	0
x_e	XDOT	-530.33	-750	-750	-750	-750	-525	-530.33
y_e	YDOT	-530.33	0	0	0	0	-535	-530.33
z_e	ZDOT	0	0	0	0	0	8	0



X CHANNEL DYNAMICS ERROR (S/N=12.5)

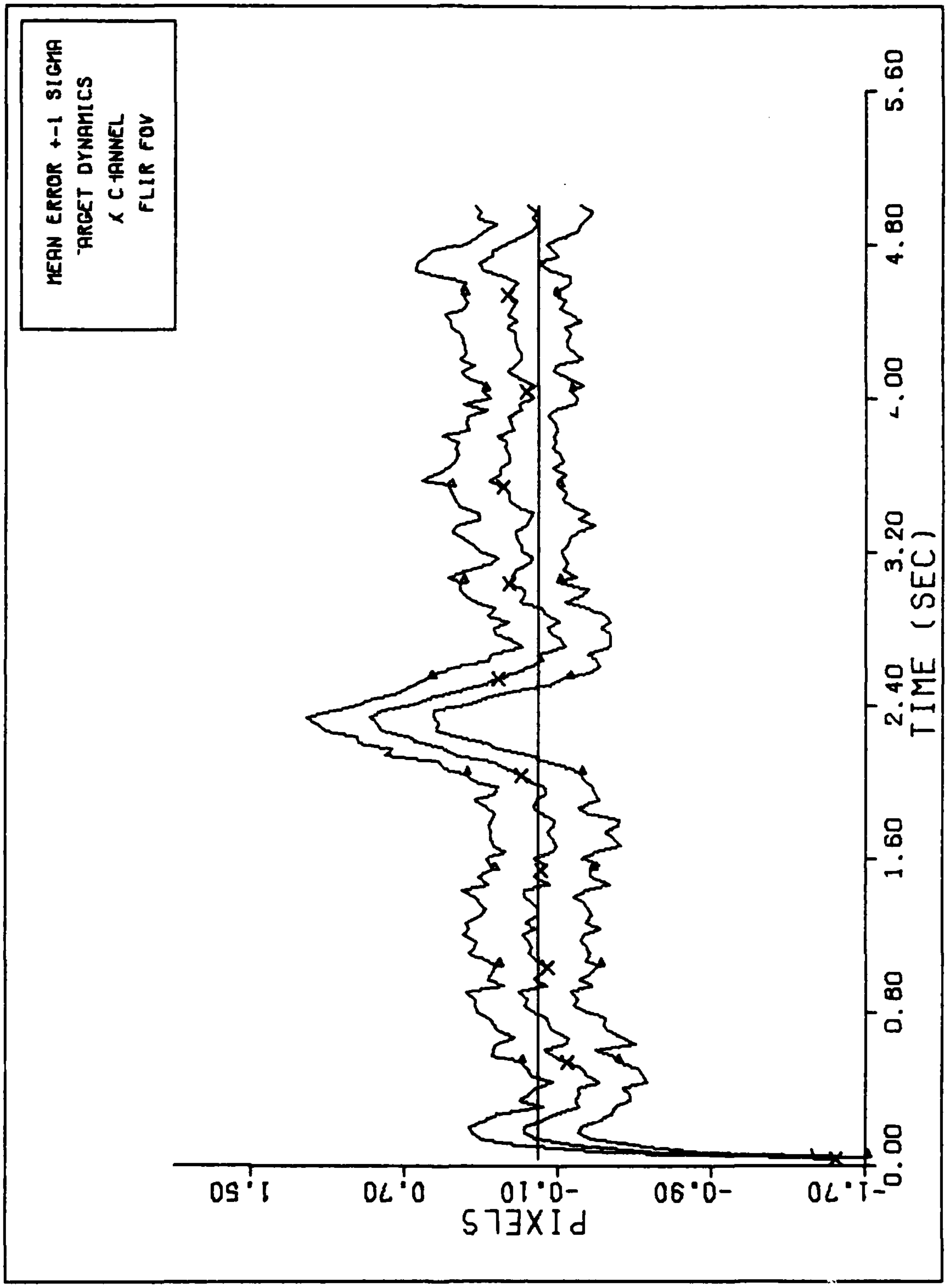
Figure G-1a

G1a



Y CHANNEL DYNAMICS ERROR (S/N=12.5)

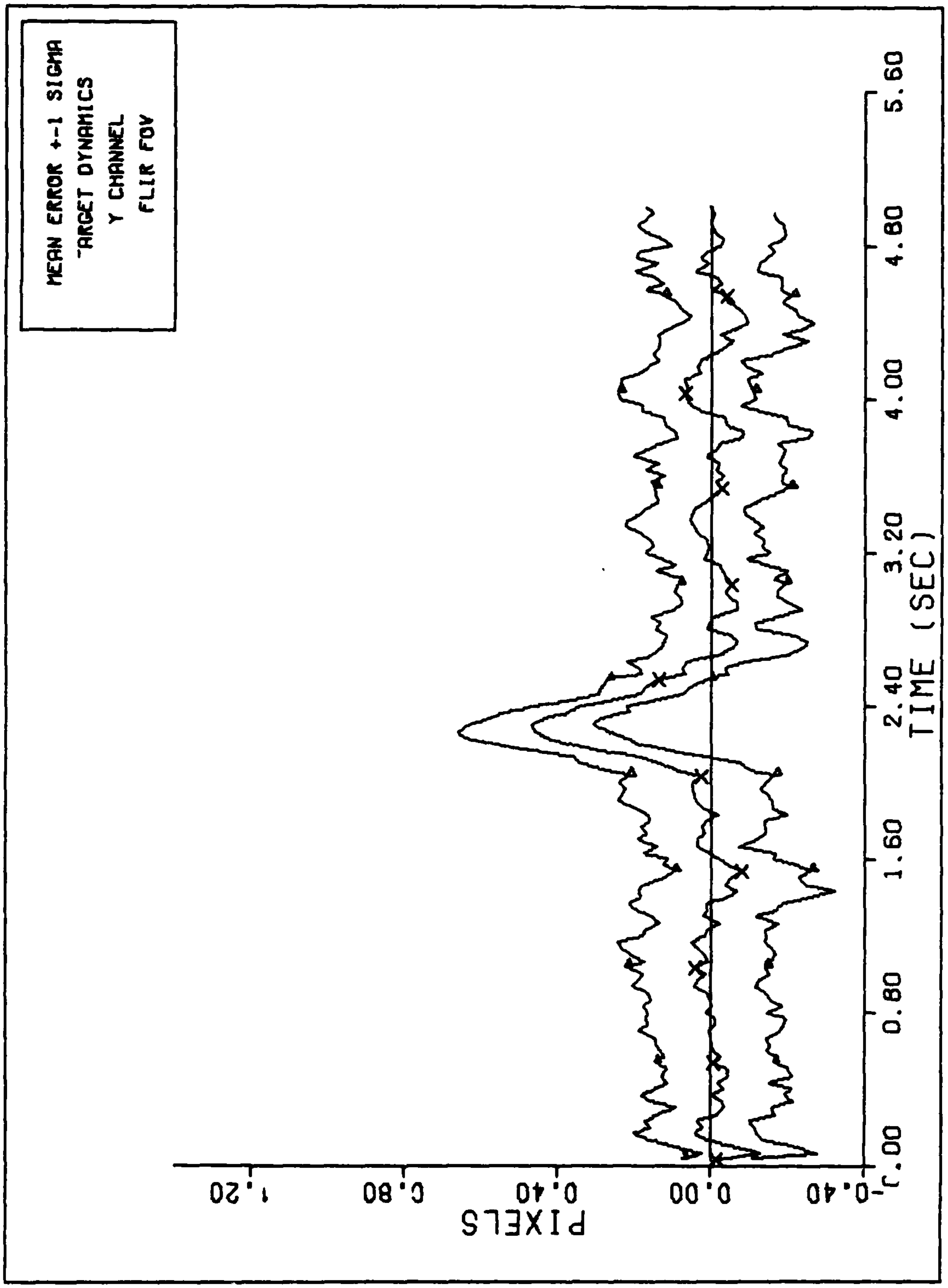
Figure G-1b



X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure G-2a

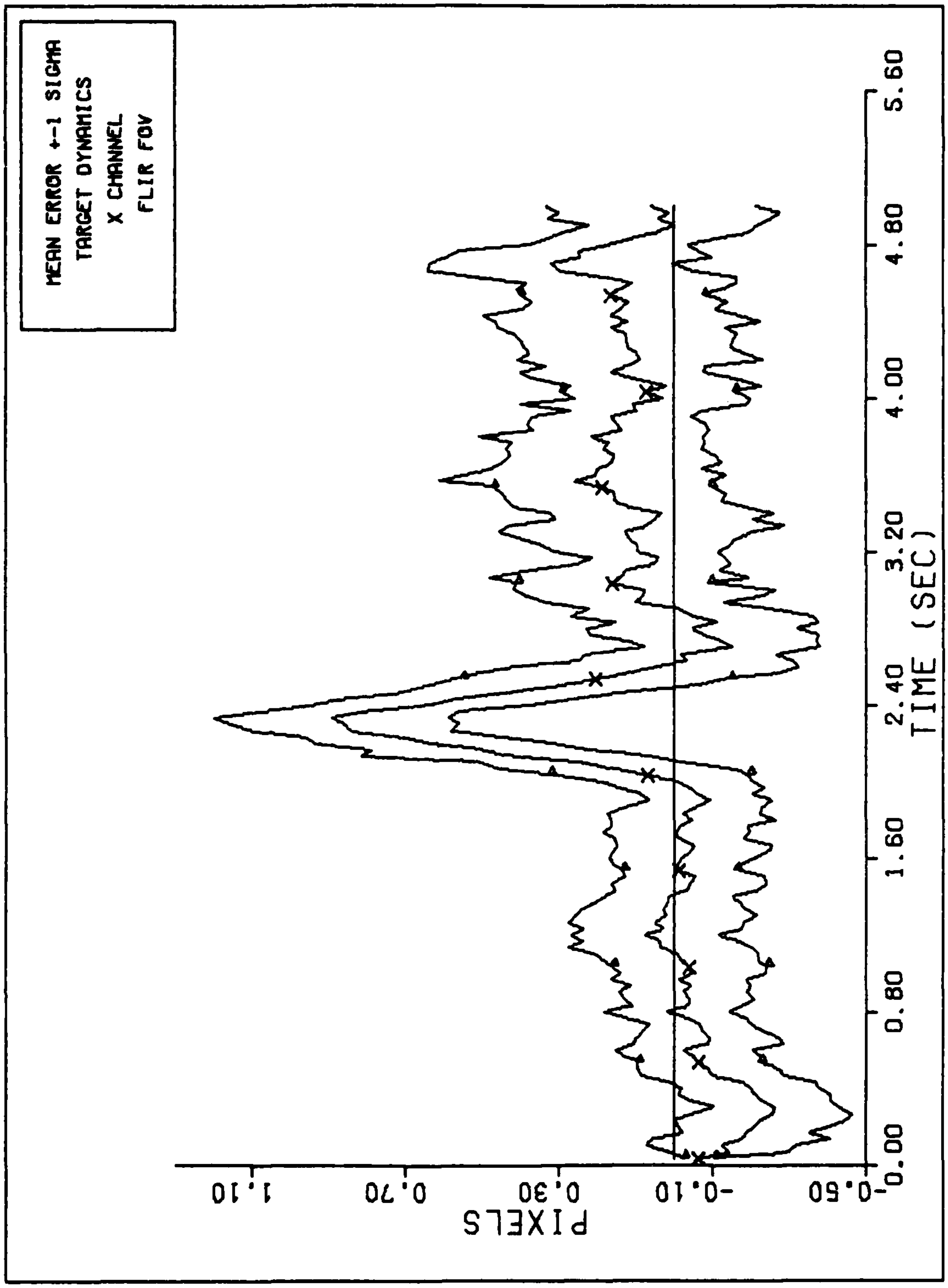
G-2a



Y CHANNEL DYNAMICS ERROR (S/N=12.5)

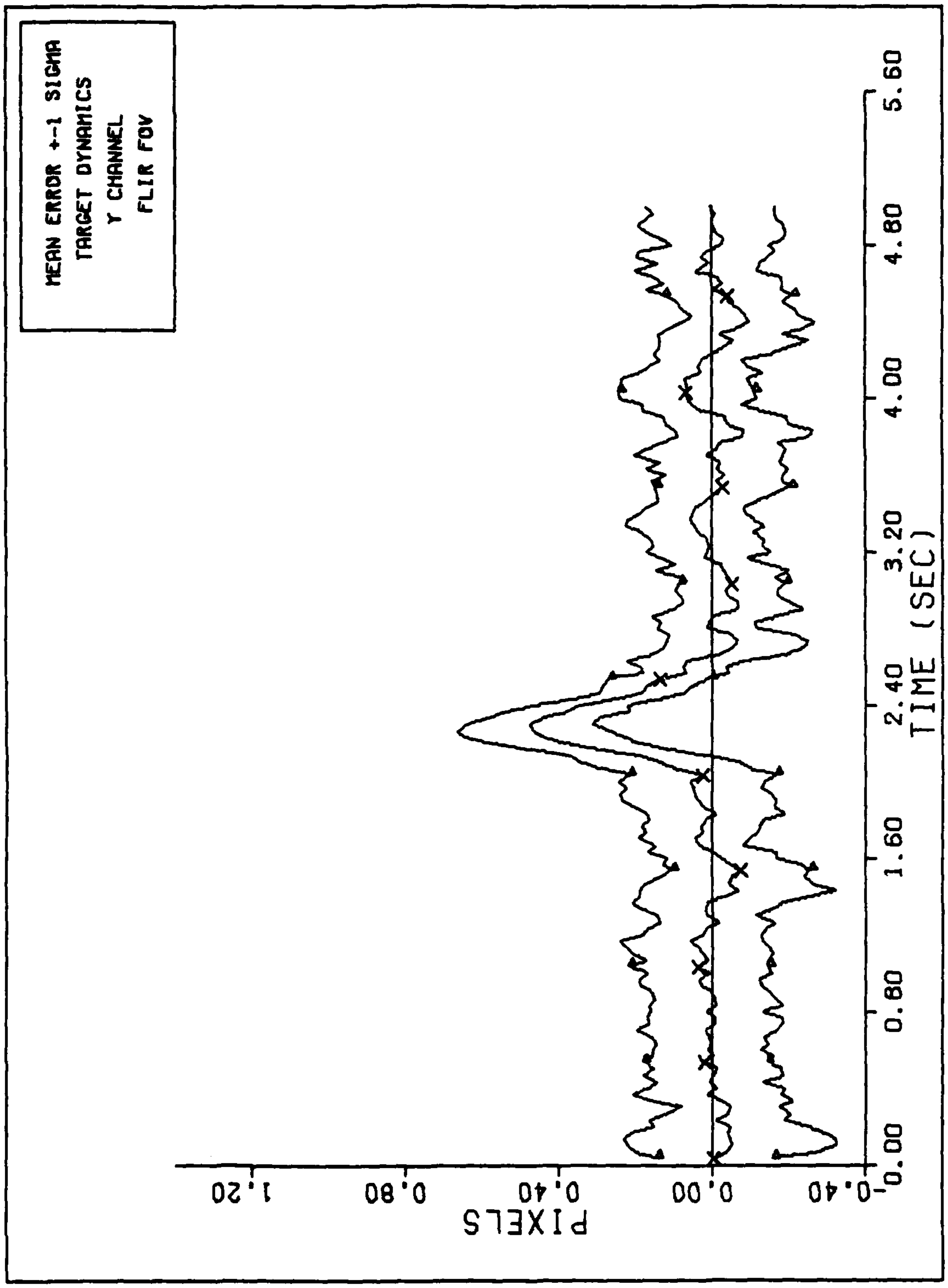
Figure G-2b

G2b



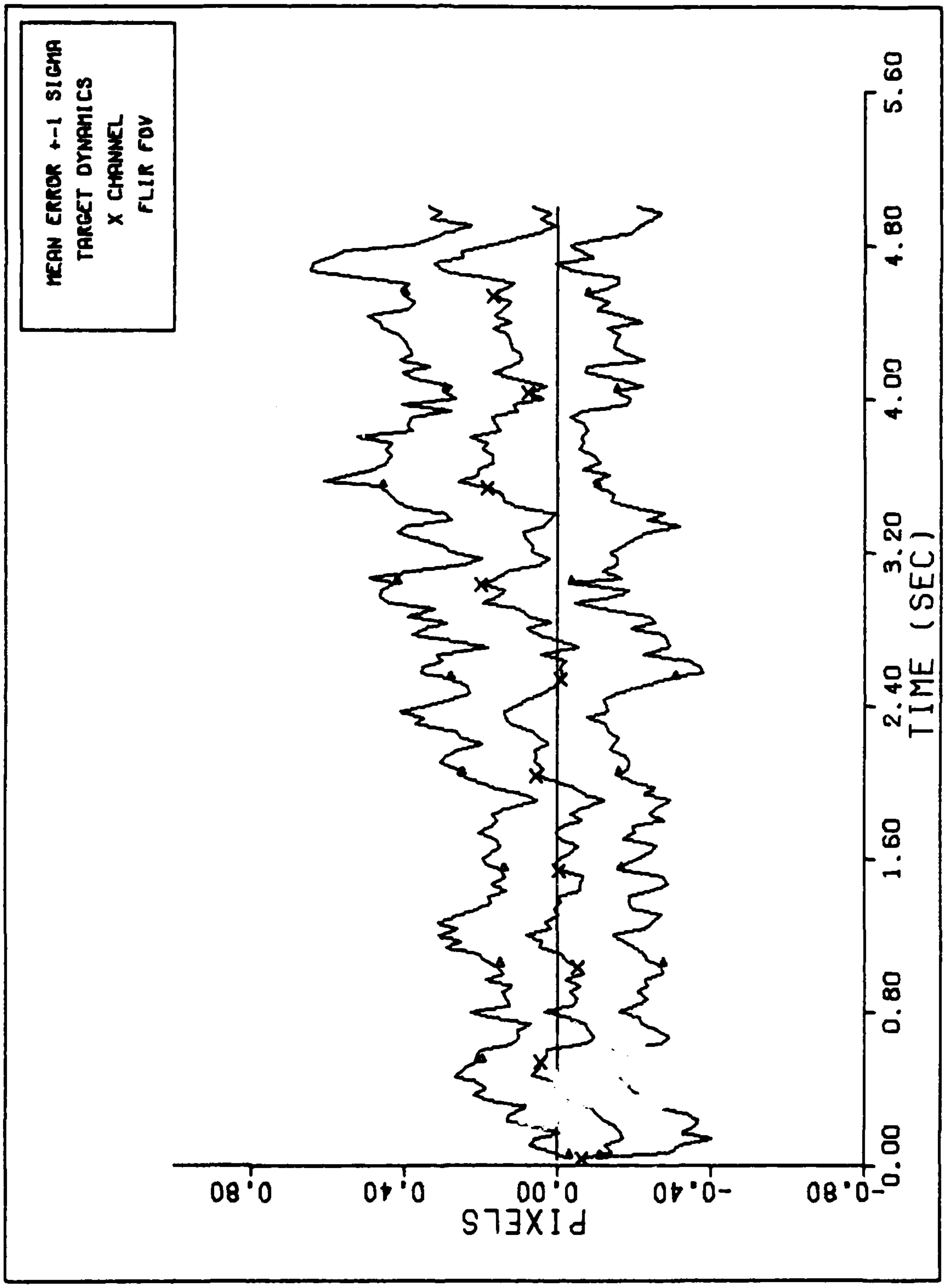
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure G-3a



Y CHANNEL DYNAMICS ERROR (S/N=12.5)

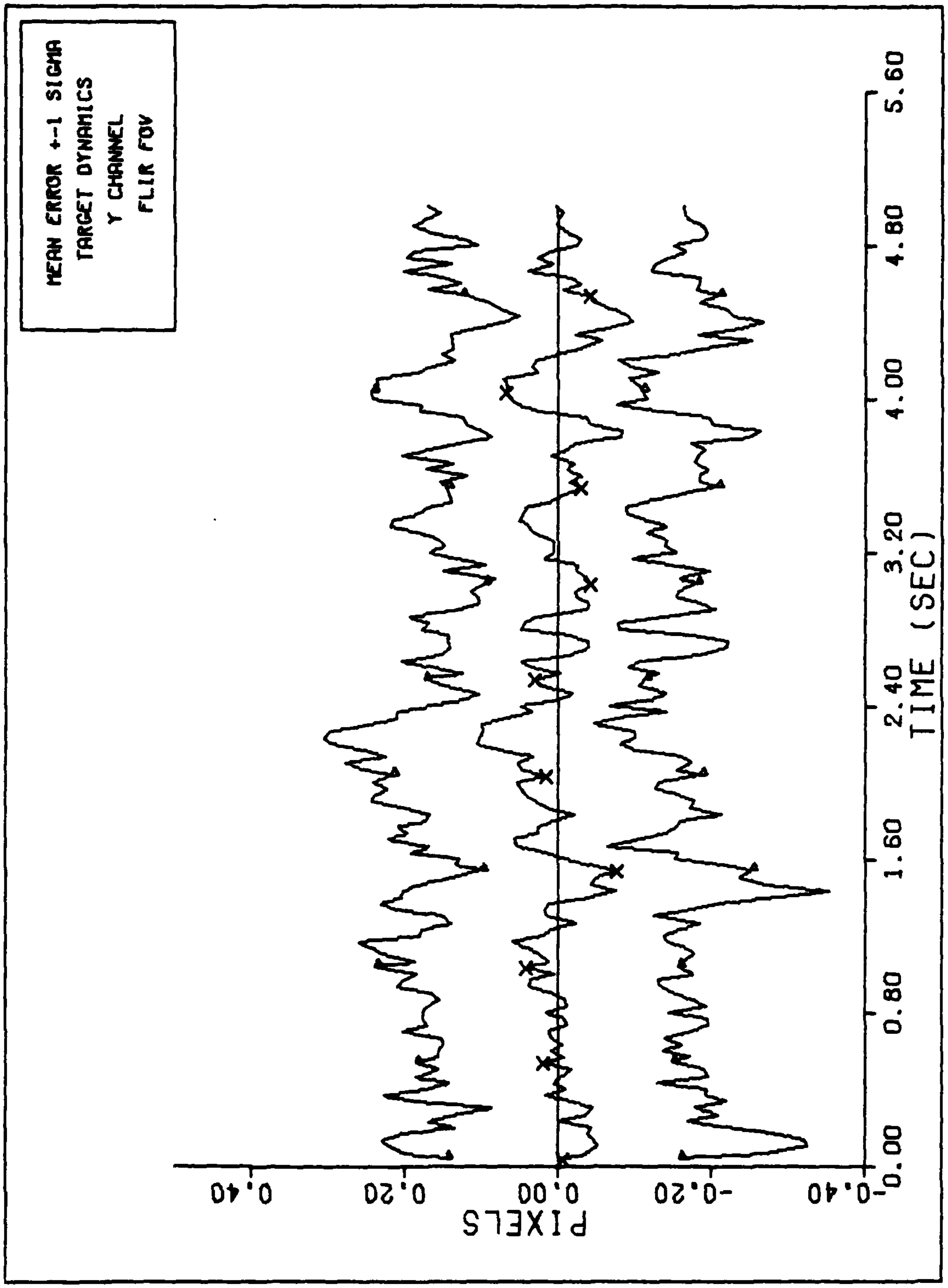
Figure G-3b



X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure G-4a

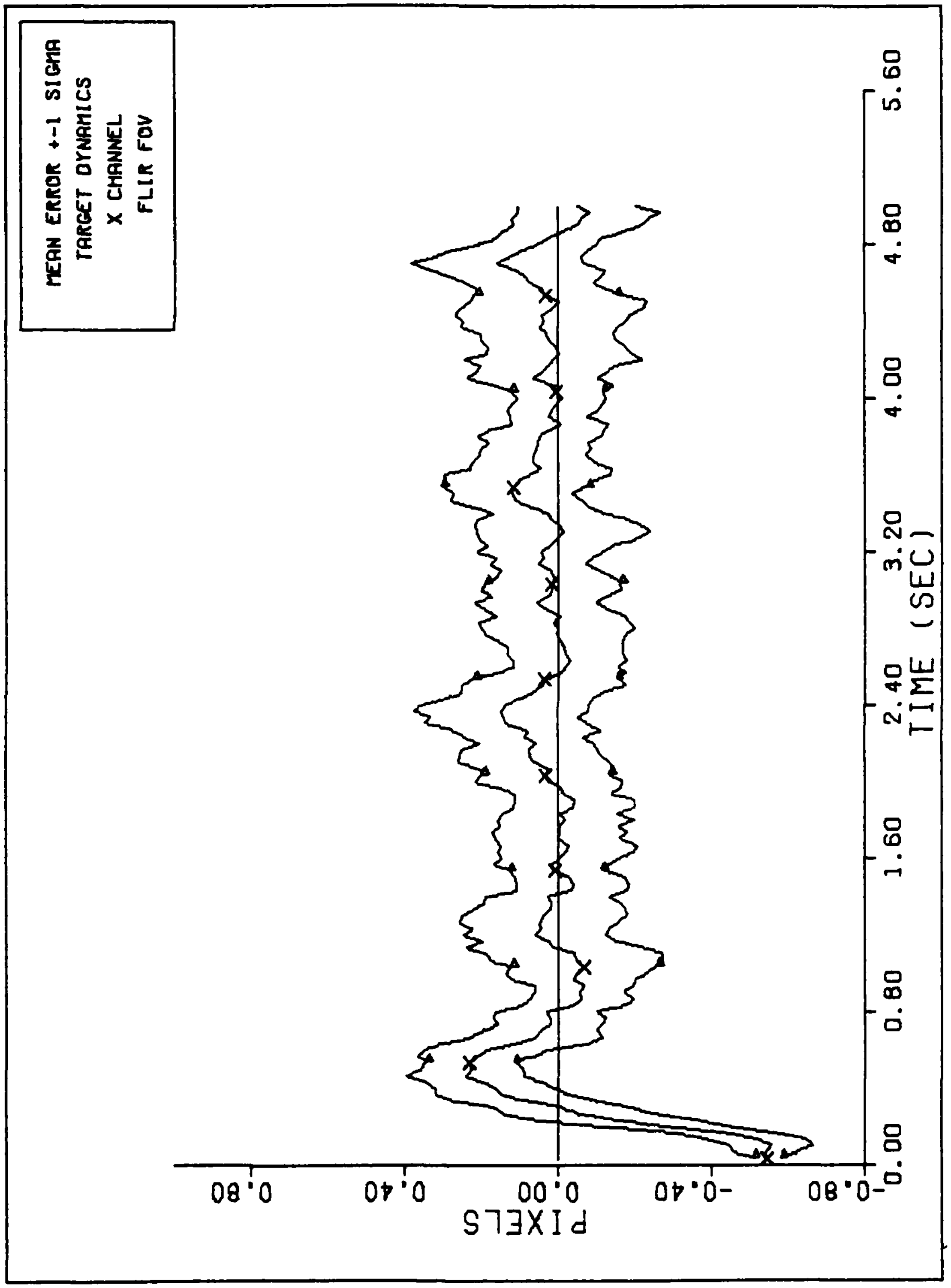
G.4a



Y CHANNEL DYNAMICS ERROR (S/N=12.5)

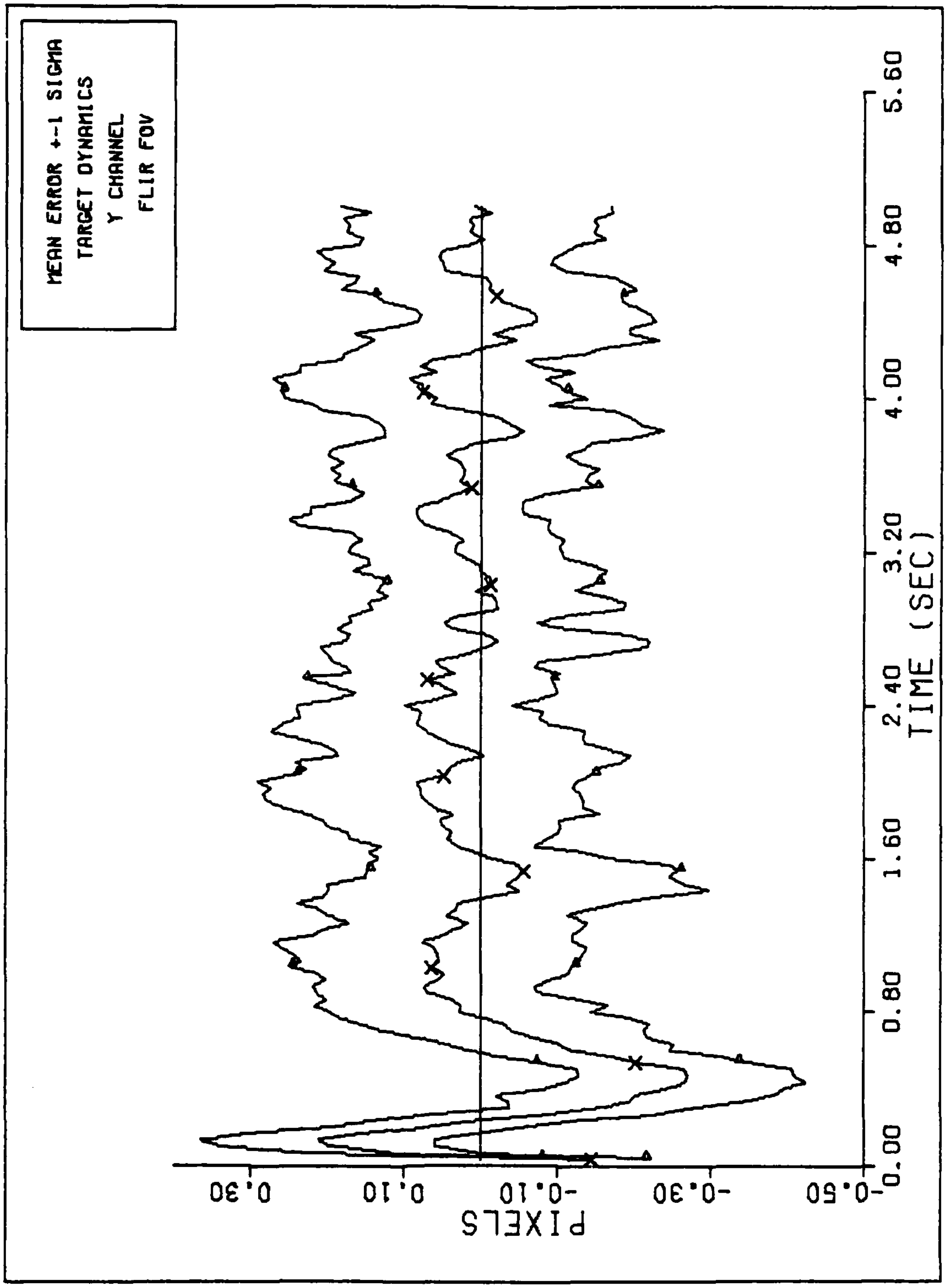
Figure G-4b

G4b



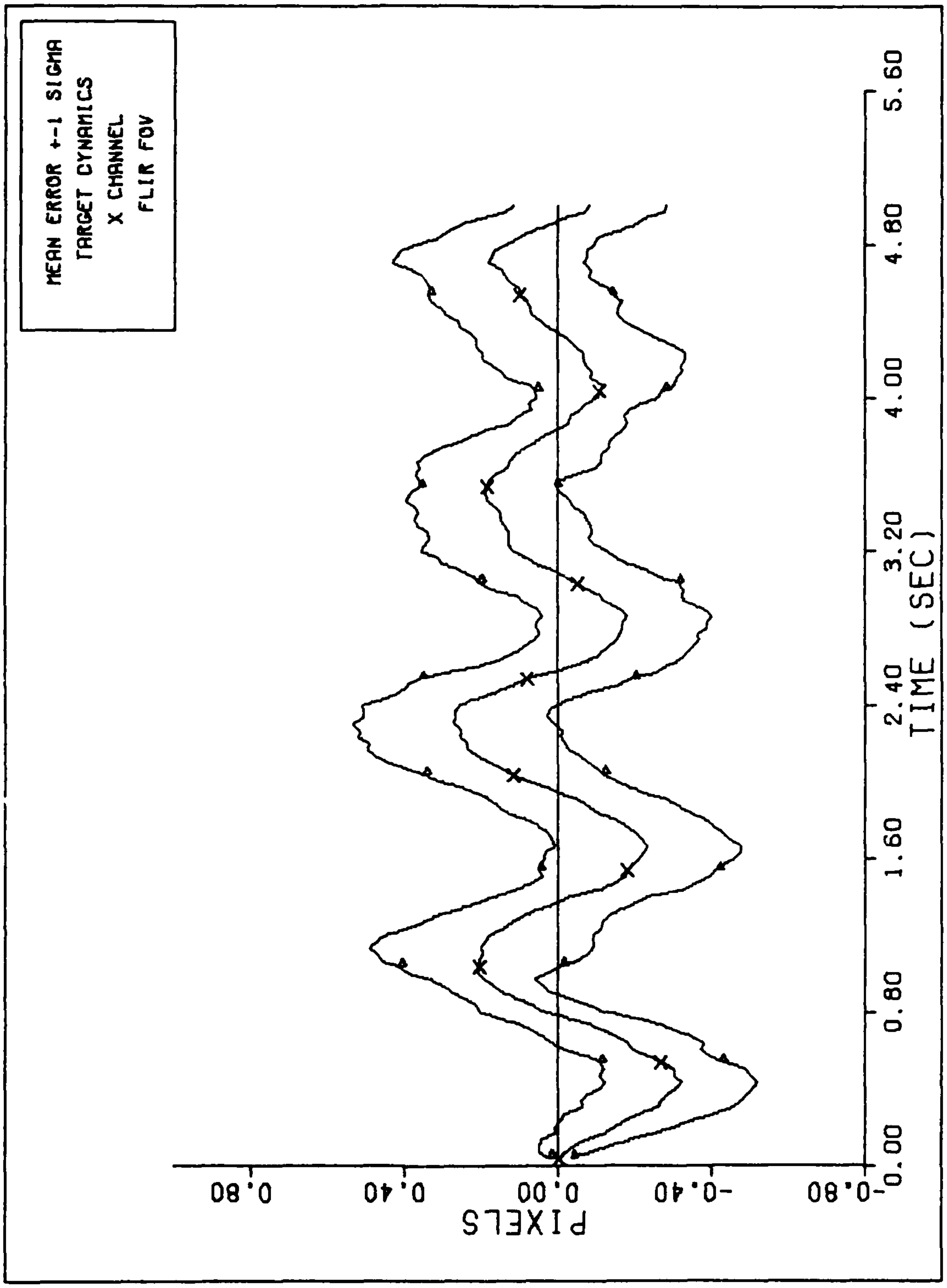
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure G-6a



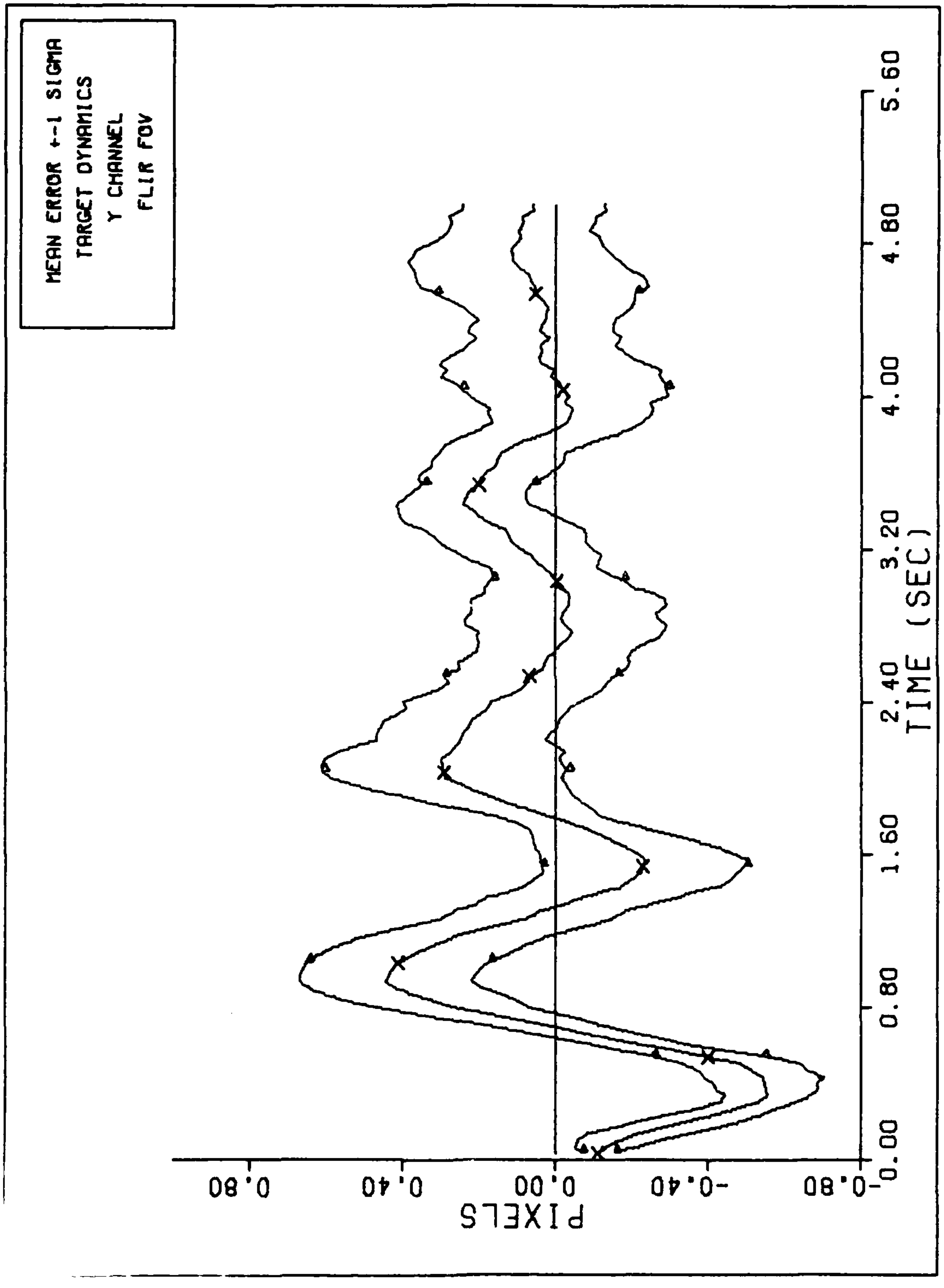
Y CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure G-6b



X CHANNEL DYNAMICS ERROR (S/N= 5)

Figure G-7a



Y CHANNEL DYNAMICS ERROR (S/N= 5)

Figure G-7b

VITA

Paul R. Leuthauser was born on June 6, 1955 in Des Moines, Iowa. He graduated from the United States Air Force Academy with a Bachelor of Science degree in Engineering Sciences and a commission in the United States Air Force in 1977. From 1977 to 1980, he served as a Flight Test Instrumentation Engineer at the Air Force Flight Test Center, Edwards AFB, California. In 1980, Captain Leuthauser was assigned to the Air Force Institute of Technology to pursue a Master's Degree in Astronautical Engineering.

Permanent address: 2200 Capitol Ave.
Des Moines, Iowa 50317

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GA/AA/81D-8	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ALTERNATIVES TO AN EXTENDED KALMAN FILTER FOR TARGET IMAGE TRACKING		5. TYPE OF REPORT & PERIOD COVERED M S Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Paul R. Leuthauser, Captain, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, OH 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of Aeronautics and Astronautics (AFIT/ENY) Wright-Patterson AFB, OH 45433		12. REPORT DATE December 1981
		13. NUMBER OF PAGES 274
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

APPROVED FOR PUBLIC RELEASE AFR 190-17.

Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

18. SUPPLEMENTARY NOTES

28 JAN 1982

FREDRIC C. LYNCH, Major, USAF

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Kalman Filter
Target Tracking
Nonlinear Filtering

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Four alternative filters are compared to an extended Kalman filter (EKF) algorithm for tracking a distributed (elliptical) source target in a closed loop tracking problem, using outputs from a forward looking (FLIR) sensor as measurements. These were 1) an EKF with (second order) bias correction term, 2) a constant gain EKF, 3) a constant gain EKF with bias correction term, and 4) a statistically linearized filter. (continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

U Estimates are made of both actual target motion and of apparent motion due to atmospheric jitter. These alternative designs are considered specifically to address some of the significant biases exhibited by an EKF due to initial acquisition difficulties, unmodelled maneuvering by the target, low signal-to-noise ratio, and real world conditions varying significantly from those assumed in the filter design (robustness). Filter performance was determined with a Monte Carlo study under both ideal and non ideal conditions for tracking targets on a constant velocity cross range path, and during constant acceleration turns of 5G, 10G, and 20G. The EKF with bias correction term demonstrated nearly identical performance to the EKF, with a cost of four times the computational burden. The constant gain EKF ideal performance was not as good, but it was more robust and required only one-tenth the computational burden. The statistically linearized filter was unsuccessful due to the particular discretization approximation used in evaluating conditional expectations inherent in its structure.

END

DATE
FILMED

3-82

DTIC